MASTER EN SCIENCES INFORMATIQUES

Analyse quantitative de cellules par traitement d'images

Yang, Xun

Award date:
1992

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix
Faculté d'Informatique
Rue Grandgagnage, 21, 5000 Namur

ANALYSE QUANTITATIVE DE CELLULES PAR TRAITEMENT D'IMAGES

Xun YANG

Mémoire présenté en vue de l'obtention du titre de
Licencié et Maître en Informatique

Promoteurs : Madame le Professeur Monique NOIRHOMME-FRAITURE
(Institut d'Informatique)

Co-Promoteurs : Monsieur le Professeur José REMACLE
(Laboratoire de Biochimie Cellulaire de la Faculté des Sciences)

SEPTEMBRE 1992

Remerciements

Nous tenons tout d'abord à exprimer notre reconnaissance à Madame le Professeur Monique Noirhomme-Fraiture, qui a accepté de diriger ce mémoire et nous a permis par ses conseils pertinents, de mener ce travail à terme.

Nous remercions Monsieur le Professeur José Remacle, le co-promoteur, de nous avoir accueillis dans son laboratoire. Sans ses conseils constructifs et sans ses critiques pertinentes, ce mémoire n'aurait pas abouti.

Nous sommes particulièrement reconnaissants pour l'aide logistique fournie par l'unité de biochimie cellulaire. Nous voudrions adresser un tout grand merci à tous ceux qui nous ont aidés pendant ce travail. Nous remercions spécialement Marc Dieu, Andrée Houbion, Martine Raes, Carine Michiels pour leur aide technique dans le domaine biochimique, leur disponibilité, et surtout leur gentillesse.

Nos remerciements vont aussi à Messieurs les Professeurs Jean-Pierre Peters, Jean-Paul Rasson ainsi que Messieurs les assistants Vincent Grandville, et Gui Huys pour les conseils judicieux donnés.

Nous remercions vivement nos amis Madame Anne Defet, Mademoiselle Rita De Meulemeester, Monsieur Laurent Goche pour la lecture de notre travail et nos amis chinois Messieurs Liming Yu et Yan He pour l'aide photographique.

Résumé

Nous avons conçu, dans le cadre de ce mémoire, un logiciel de comptage automatique de cellules vivantes en culture pour le laboratoire de biochimie cellulaire des F.U.N.D.P. Le traitement d'images comprend plusieurs filtrages d'une image digitalisée d'un champ de cellules afin d'obtenir une image suffisamment nette pour être dénombrée. Le Logiciel PCSCOPE a été utilisé à cet effet.

Dans ce mémoire, nous présentons les aspects biologiques et informatiques de l'analyse quantitative des cellules, l'objectif du travail, le principe du traitement d'images et les fonctionnalités du logiciel. Certaines caractéristiques du système sont évaluées. Les perspectives de développement du système sont également mises en évidence à la fin du mémoire.

Abstract

Within the scope of this dissertation, we have implemented a software for counting living cells in culture for the laboratory of Cell Biochemistry at the University of Namur (F.U.N.D.P). Image processing consists of turning a digital image into another image by means of a series of filtering operations. In order for a computer to count automatically the cells in the image, we should take advantage of the image processing technology to render the image neat enough. PCSCOPE, an image processing software was used for this purpose.

In this dissertation, we discuss the biological and computing aspects concerning the quantitative analysis of cells, the objective of the work, the general principle of image processing and the fonctionalities of the software. Some important characteristics of the system have been evaluated. Furthermore, further developements of the system are put into perspective at the end of this dissertation.

Table des Matières

Chapitre 1 :

Introduction

Dans le cadre d'une bonne compréhension de notre travail et des problèmes posés par l'adaptation du travail informatique au domaine biologique, nous pensons qu'il est utile de parler d'abord de l'aspect biologique de notre travail. Ensuite, nous définirons l'objectif du travail informatique.

1.1. La cellule et le vieillissement des cellules

Tous les êtres vivants sont constitués de petites unités structurales et fonctionnelles appelées cellules. Les études sur les cellules constituent à notre époque une branche importante des sciences de la biologie cellulaire. Un des secteurs de recherche consiste à suivre le vieillissement cellulaire. L'unité de biochimie cellulaire des F.U.N.D.P. mène des recherches importantes dans ce domaine.

Nous savons que chaque espèce est caractérisée par une durée de vie bien spécifique et subit un processus constant de vieillissement. Cette affirmation est également valable au niveau cellulaire. Les recherches biochimiques ont pour objectif d'évaluer l'évolution des systèmes des cellules vivantes tout au long de leur vie, que ce soit dans des conditions normales ou encore dans des conditions de stress. Un des modèles expérimentaux classiques qui conduit à ce genre d'évaluation est appelé "différenciation in vitro"

des fibroblastes humains. Ce modèle tient compte des différences qui existent entre l'état jeune et l'état vieux et apparaît comme une suite d'étapes qui se produisent au cours des sous-cultures. On peut alors envisager ensuite de tester l'effet des stress sur le passage d'une étape à l'autre [Tous 91].

En observant la morphologie des cellules que nous avons utilisées pour la mise au point de ce travail au cours de leur vie et leurs caractéristiques biochimiques sur gels d'électrophorèse à deux dimensions réalisés à partir des protéines cellulaires, on peut identifier sept étapes: les cellules en culture in vitro peuvent être classées en sept types (voir Figure 1-1) dont les proportions varient suivant l'âge de la culture. Dans notre cas, nous ne nous intéressons qu'aux trois premiers types:

Type I: représente la majorité des cellules dans une culture de générations 0 à 20. Les cellules de type I sont en forme de fuseau très effilé et elles sont souvent très réfringentes en microscopie de phase. Leur contour est très régulier.

Type II: représente la majorité des cellules dans une culture de générations 20 à 40. Elles sont d'aspect plus rectangulaire et plus ramassé.

Type III: représente la majorité des cellules dans une culture après la génération 40. Elles sont de formes plus variées; leur taille est plus importante que celle des cellules de type II et leur pourtour est un peu effiloché.

Ces trois types sont appelés mitotiques, c'est-à-dire que les cellules sont encore capables de se diviser en culture.

Les quatre **types IV, V, VI, et VII** sont appelés postmitotiques, parce que les cellules ne sont plus capables de se diviser. Elles se trouvent en quantité faible en culture dans des conditions normales. La cellule de type IV est très allongée. Les types V et VI sont représentés par des cellules très grandes au contour irrégulier, avec un cytoplasme désorganisé. Le type VII est une cellule en dégénérescence.

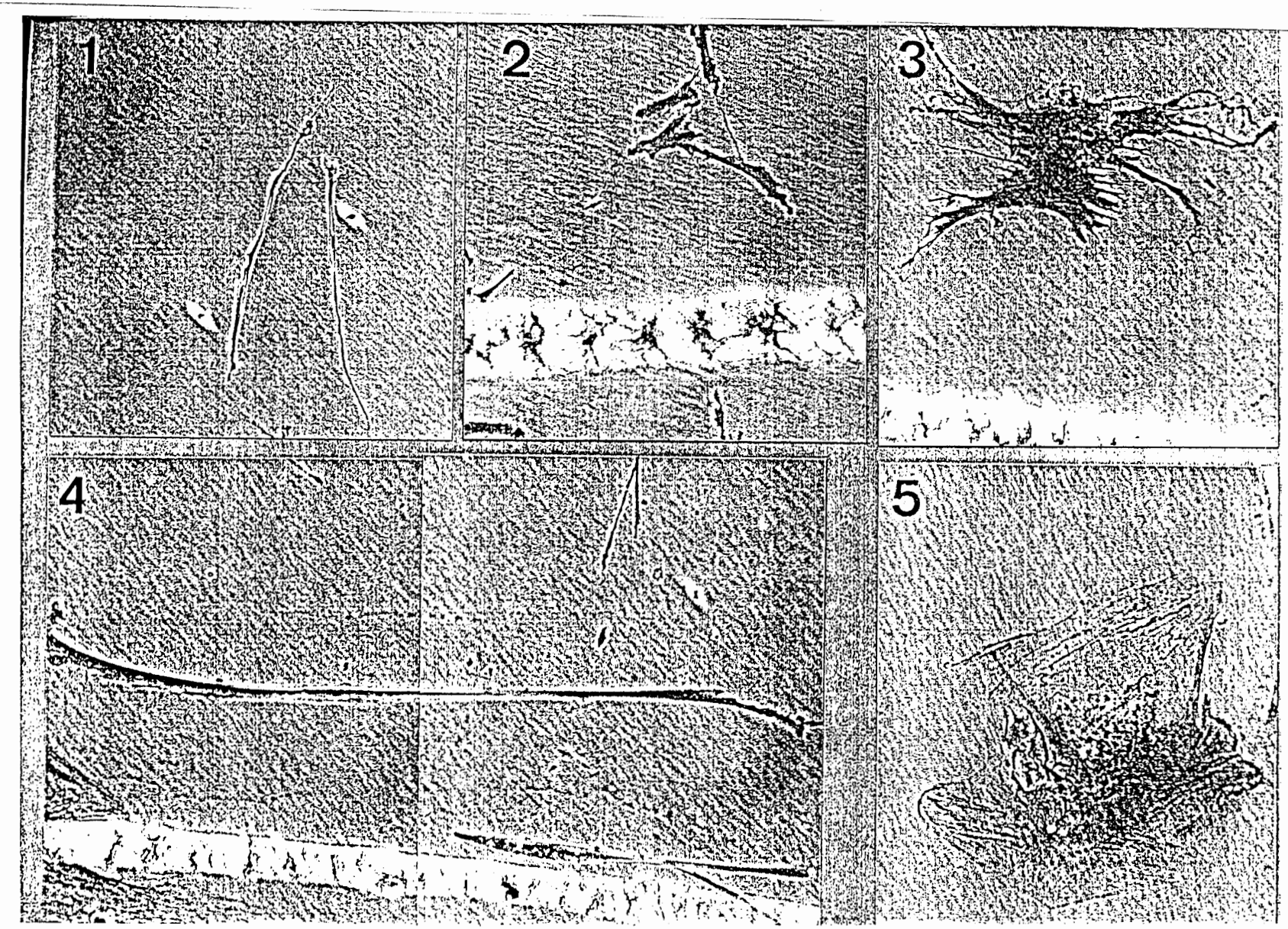
1
2
3
4
5

Figure 1-1 Les différents types morphologiques des fibroblastes WI-38. *in vitro* Type I, II, III pour les cellules mitotiques et type IV à VII pour les cellules post-mitotiques. Coloration des cellules au Bleu de Coomasie après fixation. Grossissement 250x.

1.2 L'analyse quantitative des cellules

Comme nous l'avons dit ci-dessus, le modèle de différenciation se base sur l'observation morphologique et l'analyse biochimique.

Notons que notre travail informatique est particulièrement lié à l'observation morphologique. L'analyse biochimique met en oeuvre des techniques d'analyse sophistiquées qui sont basées sur l'analyse des constituants des cellules, c'est-à-dire, principalement les enzymes, le DNA (Deoxyribonucleic acid) ou diverses structures subcellulaires comme les mitochondries, les lysosomes ou les membranes. Au cours de ces analyses, certains programmes informatiques peuvent être utilisés comme ce fut le cas pour l'analyse des protéines après électrophorèse en gel à deux dimensions par V. Hénin [Héni 89].

L'intérêt d'avoir à sa disposition un modèle de vieillissement des cellules bien caractérisé est qu'il est possible d'étudier l'effet de molécules diverses afin de voir si elles ont un effet protecteur ou au contraire un effet stressant pour les cellules. Comme au cours de l'évolution spontanée "vieillissement", les cellules passent d'un type cellulaire à l'autre, il est possible d'examiner si les stress affectent ce passage. Pour ce faire, il faut déterminer le nombre de cellules des divers types puis refaire la même détermination après le stress et examiner s'il y a eu un effet.

1.3 L'objectif du mémoire

La collaboration entre les traitements informatisés et non-informatisés dans le domaine biomédical met en place un environnement très fascinant pour les informaticiens.

Traditionnellement, les biochimistes utilisent des microscopes pour analyser et étudier la morphologie des cellules en culture, tandis que les informaticiens bénéficient d'un matériel performant. Dès lors, ceux-ci pourraient aider ceux-là et les libérer d'un travail harassant et routinier.

Nous avons vu qu'en observant la morphologie des cellules à l'aide d'un microscope, les biochimistes procèdent à certaines analyses quantitatives. Une de ces analyses quantitatives consiste à compter des cellules dans des conditions normales ou dans des conditions de stress.

En effet, les cellules attachées sur le support (boîte de Pétri) sont comptées chaque jour et sont considérées comme cellules vivantes. Michiels et al.[Mich.90], en utilisant le test de viabilité basé sur le principe d'exclusion d'un colorant (acridine orange-bromure d'éthidium) [Park 79], a montré que ces cellules étaient bien vivantes.

Il est évident que l'observation des cellules et le comptage sous microscope sont des tâches très fastidieuses et qui prennent du temps, à fortiori, puisque le nombre de cellules à observer est important ainsi que les nombreuses cultures à examiner. Nous espérons, dans ce travail, grâce à l'analyse au microscope et une caméra couplée à un ordinateur, augmenter le nombre de cellules comptées et ceci dans un temps plus court.

Le problème posé est donc de savoir s'il est possible de faire ces comptages automatiques de cellules dans ces conditions.

Vu la spécificité du problème décrit ci-dessus, l'Unité de Biochimie Cellulaire des F.U.N.D.P souhaitait voir développer un système informatique qui pourrait acquérir et digitaliser automatiquement les images de cellules, et les compter. Ceci va constituer l'objectif de ce mémoire.

En coopération avec les biochimistes, nous nous intéressons principalement à la partie informatique. Donc, le travail présent est une première approche d'une analyse quantitative de cellules. Nous tentons de compter automatiquement les cellules à l'aide de traitements d'images digitalisées.

Le comptage automatique des cellules lui-même n'est pas un nouveau problème informatique. Normalement, afin de réaliser le comptage de cellules avec l'ordinateur, on doit adapter successivement les trois étapes générales suivantes [Vinc 87].

La première est la digitalisation des images de cellules vues au microscope. Cette étape s'appelle aussi pré-traitement d'images et constitue une étape préliminaire. La deuxième est le traitement d'images proprement dit. Cette étape s'effectue afin de rendre les images microscopiques digitalisées plus adaptées à l'analyse quantitative que l'on souhaite faire. La dernière est le comptage des cellules sur les images traitées à l'étape précédente.

Le comptage automatique des cellules dans les systèmes existants [Land 84] [Lest 78] [Mich 87] s'effectue sur des cellules colorées. Dans notre cas, les cellules sont toujours vivantes et

maintenues en culture. On ne peut pas les colorer pour augmenter les contrastes afin de les conserver in vitro pendant plusieurs jours. Cette contrainte rend les images digitalisées moins nettes (voir figure 1-2), et rendra par conséquent notre travail plus difficile. Il faut aussi signaler le problème posé par les contours très variables des cellules et des tailles très différentes. Ces contraintes expliquent la difficulté de ce travail comparé au comptage classique des globules rouges par exemple. Nous présentons à la figure 1-2 une image digitalisée d'une culture de cellules prise au grossissement 36x du microscope. On remarque bien les formes diverses avec des ramifications très importantes. Il est possible aussi que les cellules s'agrègent. Dans ce cas-là, le comptage des cellules avec l'ordinateur est inévitablement très subjectif.

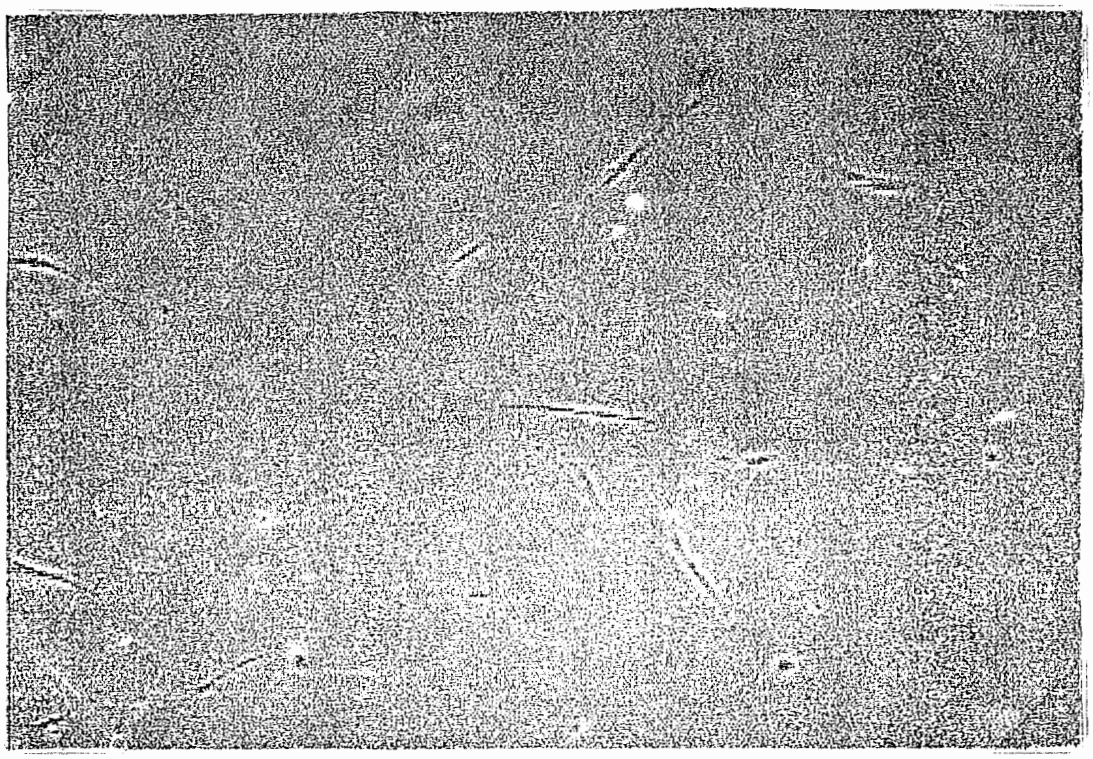

Figure 1-2 Une image digitalisée de cellules

En tenant compte de l'objectif de notre travail et au vu de la technologie du traitement d'images, nous avons pu réaliser un système qui permet de compter automatiquement des cellules attachées dans une boîte de Pétri.

Le reste du mémoire s'organise comme suit. Dans le chapitre suivant, nous parlerons des ressources informatiques et biologiques dont nous disposons. Dans les sections 3, 4 et 5, nous allons exposer le fonctionnement du système que nous avons réalisé pour le comptage des cellules en culture. Dans ces chapitres, nous allons également décrire la méthodologie des techniques du traitement d'images, et les algorithmes d'analyse quantitative sur les images traitées. Le dernier chapitre est consacré à faire un bilan du travail. Certaines caractéristiques du système y seront évaluées.

Chapitre 2 :

Matériel et logiciels

Avant de développer le système permettant l'analyse et le comptage des cellules, il nous semble nécessaire de préciser les ressources biologiques et informatiques disponibles.

2.1 Matériel biologique et méthode de culture des cellules

Les cellules fibroblastes W1-38 utilisées proviennent de poumon d'embryon humain. Les fibroblastes WI-38 ont été cultivés dans un milieu BME (Flow Laboratories, Angleterre) contenant 10% de sérum de bovin foetal (Gibco, Ecosse) dans des boîtes (voir figure 2-1) de Pétri en plastique d'un diamètre de 6 cm (Falcon, Angleterre) et dans une atmosphère d'air humide à 95% contenant 5% de CO_2 et à 37 °C.

La mise en culture et la manière dont les cellules sont cultivées, c'est-à-dire les repiquages, sont décrites à la figure 2-1 [Hayf 65]:

Le repiquage des cellules se fait lorsque les cellules en se divisant ont atteint la confluence, c'est-à-dire qu'elles se touchent les unes les autres et ainsi ne savent plus se diviser. Le repiquage consiste alors à augmenter la surface de culture afin que les cellules puissent continuer de se diviser. Pour ce faire, les cellules sont détachées de leur support puis réparties dans deux ou plusieurs flacons de même surface, de la manière suivante:

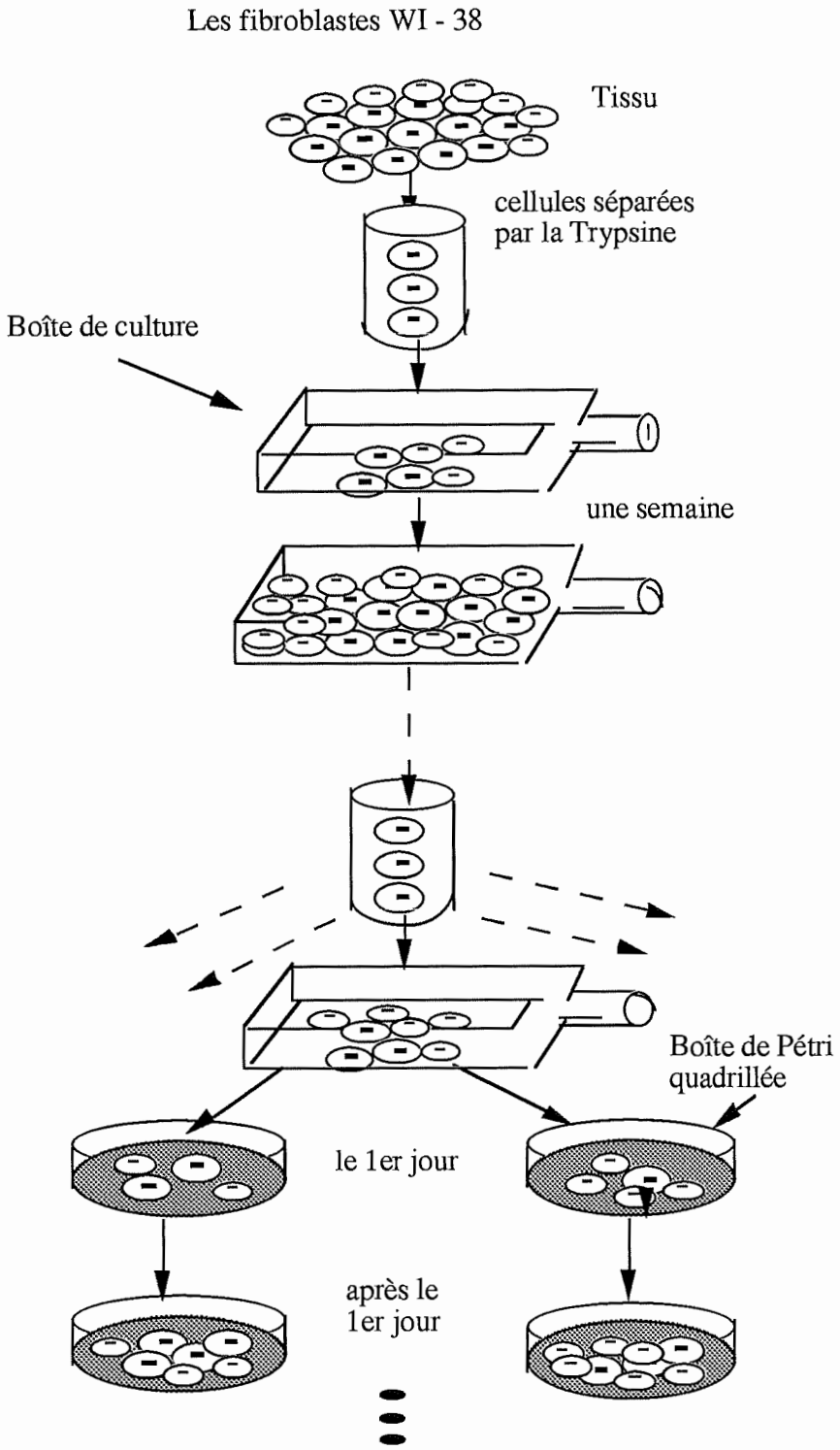

Figure 2-1 Mise en culture de fibroblastes à partir d'un tissu et techniques de repiquage. Les fibroblastes WI-38 se divisent entre 40 et 60 fois. Leur capacité de division diminue avec les repiquages successifs. La congélation n'affecte en rien leur nombre de divisions. Cette méthodologie pour l'étude du vieillissement cellulaire fut développée par Hayflick (1965).

a) préparation: Il est nécessaire de préchauffer dans l'étuve à 37 °C le milieu seul (M), le milieu contenant 10 % de sérum (M.S) et la trypsine 0,25% (provenant de la firme GIBCO Angleterre). La hotte où se fait la manipulation est nettoyée à l'alcool et tous les ustensiles - (les bouteilles, pipettes, ...) doivent être stérilisés, afin d'éviter toute contamination.

b) repiquage: Le milieu de culture est décanté, les cellules sont rincées avec un milieu M puis traitées avec de la trypsine. Celle-ci est utilisée pour imprégner la couche de cellules et agir sur les protéines de la membrane plasmique, ce qui a pour résultat de détacher les cellules de leur support et de les séparer les unes des autres.

Les cellules ainsi détachées sont remises dans du milieu de culture (M+S) et réparties dans les flacons adéquats. Quelques gouttes de cellules sont prélevées et placées dans des boîtes de Pétri de 6cm de diamètre préalablement quadrillées à l'aide d'un diamant (voir la figure 2-2). On peut ainsi repérer les cellules comptées d'un jour à l'autre.

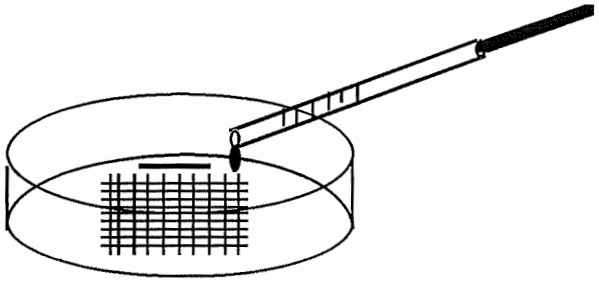

figure 2-2 Le quadrillage de boîte de Pétri

2.2 Matériel informatique (voir Figure 2-3):

- Un microscope de marque **Leitz Labovert** (microscope inversé pour l'analyse d'objets transparents). Les paramètres de grossissement du microscope sont:

oculaire: 10 fois;
objectif: 6 fois;
lentille entre le microscope et la caméra: 0,63x;
grossissement final: 10 x 6 x 0,6 = 36 fois.

- Une **iVC500 caméra vidéo CCD** (PN 562BC SN554, i2$, fabriquée en France) noir/blanc.

La caméra doit être alimentée en **J4-16** par une tension positive de 12V.

La caméra est montée sur le microscope pour l'acquisition des images de cellules. Les images acquises sont digitalisées. Les images digitalisées ont une taille maximale de 512 x 512 points (pixels). Les niveaux d'intensité lumineuse (niveau de gris) varient de 0 (noir) à 255 (blanc). Chaque point de l'image est codé sur 1 byte. Donc, la mémorisation d'une image complète nécessite un minimum de 256KB ((512 x 512 x 1byte)/1024 bytes)) de mémoire.

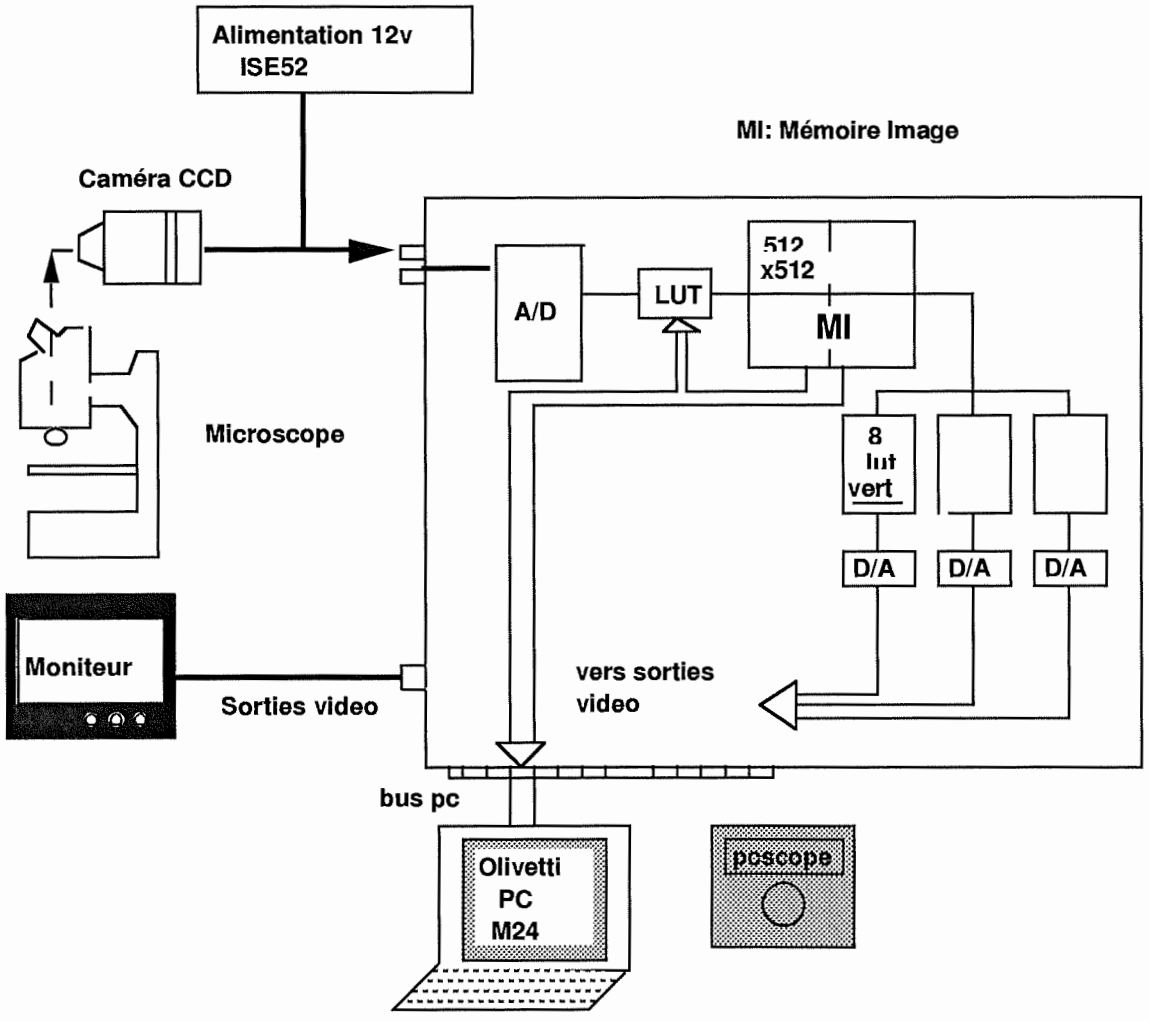

Figure 2-3 La configuration du matériel

- Une carte vision "Mémoire Image" (MI), contenant 4 images de 512 x 512 points et capable de gérer 256 couleurs ou niveaux de gris.

- Un Olivetti PC M24 avec un disque dur 20M bytes.

- Un moniteur SONY pour la visualisation des images.

2.4 Logiciels:

Un logiciel de traitement d'images PCSCOPE est disponible dans le laboratoire de Biochimie. Ce logiciel est un logiciel puissant de traitement d'images. Il fournit une bibliothèque de fonctions assez riches (écrites en langage C), notamment:

- les fonctions d'acquisition d' images,

- les fonctions de traitement d'images: filtrage, Erosion/Dilatation, etc...,

- les fonctions graphiques et alphanumériques,

- les fonctions de mémorisation des images sur disque,

- les fonctions de contrôle de la carte d'images,

- les fonctions de manipulation des LUT (Look Up Table),

- les fonctions de gestion de la souris,

- quelques fonctions spécialisées.

Malgré sa puissance de traitement d'images, PCSCOPE ne permet pas, tel quel, le comptage de cellules. Nous verrons dans ce mémoire comment PCSCOPE peut servir de base pour développer l'application qui nous intéresse.

Chapitre 3 :

Fonctionnalité et architecture du système

3.1 Microscopie

Avant de décrire les fonctionnalités du système, nous considérons qu'il est nécessaire de préciser certains termes.

Une boîte de Pétri est divisée en plusieurs champs (voir figure 3-1). Chaque image digitalisée à traiter correspond à un champ de cellules.

Un champ dans une boîte de Pétri peut être repéré en numérotant les lignes et les colonnes. Chaque champ est identifié par un couple dont le premier élément est le numéro de ligne et le second le numéro de colonne. Par convention, nous commençons à compter lignes et colonnes à partir de (0, 0). La première ligne (trait de repère) est située en haut. La première colonne est à gauche. Cela nous permet de sélectionner le champ dont l'image est à traiter. Il est évident qu'un champ contient plusieurs cellules et qu'une cellule est représentée par plusieurs pixels.

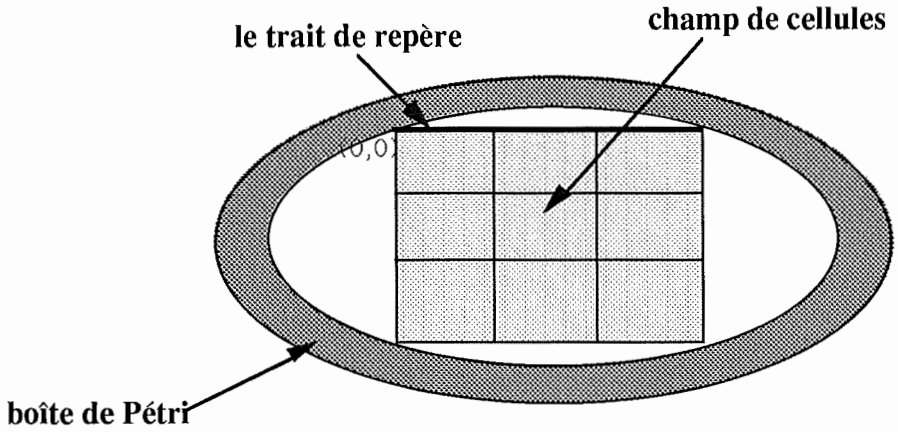

Figure 3-1 La boîte de Pétri

Comme nous l'avons dit dans la section 2.2, une image digitalisée est composée de pixels. Chaque pixel est une fonction à trois paramètres, c'est-à-dire: **F(x, y, gris).**

où: **x, y** : sont des coordonnées à deux dimensions.
gris : représente le niveau de luminosité dans une image noir et blanc.

3.2 Les fonctionnalités

Pour réaliser l'objectif décrit dans l'introduction, nous considérons que le système devrait offrir les fonctionnalités suivantes (voir Figure 3-2.). Les fonctions 1), 2) et 3) suivantes s'appliquent à une image qui correspond à un seul champ de cellules. Les fonctions 4) et 5) sont considérées du point de vue du système global:

1) Acquisition et digitalisation de l'image. Cette fonction a pour but d'obtenir les renseignements concernant une image déterminée.

On note que pour chaque champ de cellules concerné, cette fonction doit pouvoir fournir les renseignements suivants:

- l'identificateur d'une image digitalisée;

- le numéro d'identification du champ ;

- le nom de la boîte à laquelle appartient le champ de cellules dont le nombre est à déterminer;

- la date du traitement.

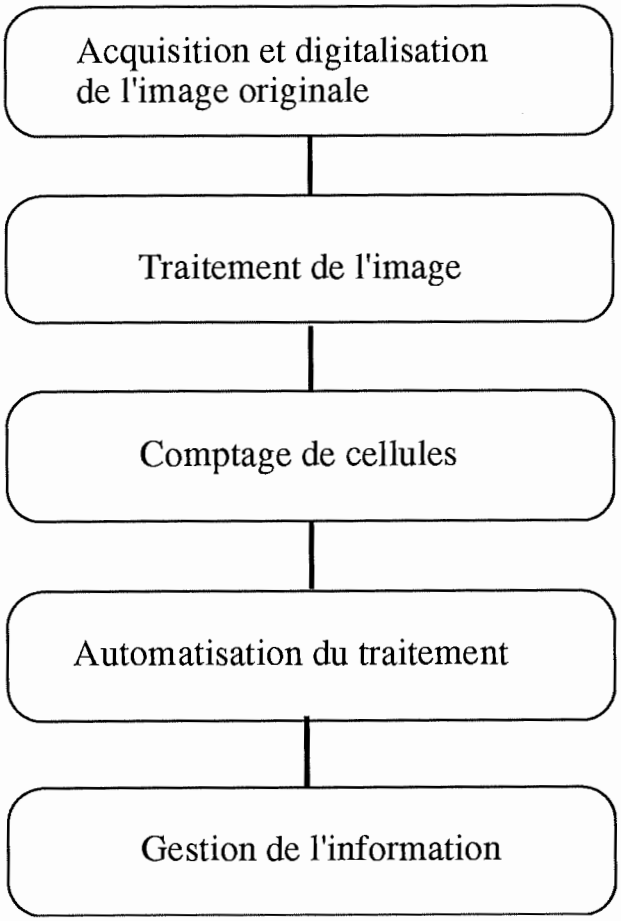

Figure 3-2 Les fonctionnalités du système

2) Traitement de l'image. Cette fonction transforme l'image d'une forme à l'autre afin de rendre le comptage de cellules plus facile.

3) Comptage de cellules présentes dans le champ. C' est le but du système. Il doit permettre de déterminer le nombre de cellules dans une image traitée.

4) Automatisation du traitement des images. Le traitement de l'image a souvent besoin d'une série de procédures pour atteindre un résultat satisfaisant. Cela veut dire que pour une image donnée, le temps de traitement est très élevé. Il est donc presque impossible de faire un comptage en temps réel avec le matériel informatique dont nous disposons.

Pour que le système ait une valeur pratique et présente un traitement standard complet, nous pensons opérer l'automatisation en deux parties.

La première est l'automatisation de l'acquisition des images. Celle-ci consiste à saisir plusieurs images et à les stocker dans le disque. (pour plus d'informations, voir la section 4.1).

La deuxième est l'automatisation du traitement et du comptage des cellules. Celle-ci peut se faire sur les images stockées en mode "batch", c'est-à-dire qu'on peut laisser l'ordinateur faire ce travail pendant la nuit.

5) La gestion de l'information. Cette fonction coordonne et réalise les fonctionnalités du système via une interface adéquate. Autrement dit: le système doit posséder une interface qui présente de façon adéquate les fonctionnalités du système à l'utilisateur.

3.3. L'architecture logicielle

Afin de réaliser les fonctionnalités décrites dans la section 3.2, nous proposons dans cette section l'architecture logicielle du système. Les différents modules du système, ainsi que les relations logicielles entre les modules sont représentés à la figure 4-1.

Le module d'**Acquisition des données** permet à l'utilisateur de saisir les images pour un ou plusieurs champs de cellules dans une boîte de Pétri.

Le module de **Traitement d'images** permet de traiter les images acquises afin de rendre la quantification des cellules facile.

Le module de **Comptage des cellules** permet de compter rapidement les cellules d'une image (un champ de cellules).

Le module de **PCSCOPE** offre des services de base relatifs au traitement d'images à usage général, c'est-à-dire, indépendant d'une application particulière.

Le module d'**Interface interactive** joue le rôle de coordinateur en appelant les fonctions des différents modules correspondant aux sélections de l'utilisateur. Il réalise également l'utilisation de la **carte-vision** lors de la mise en marche du système.

Le module d'**Accès-extra BD** permet l'accès et l'extraction de données de la BD (Base de Données). Ce système est indépendant d' un SGBD (Système de Gestion de Base de Données) particulier. Ici, BD signifie l'ensemble de fichiers disques.

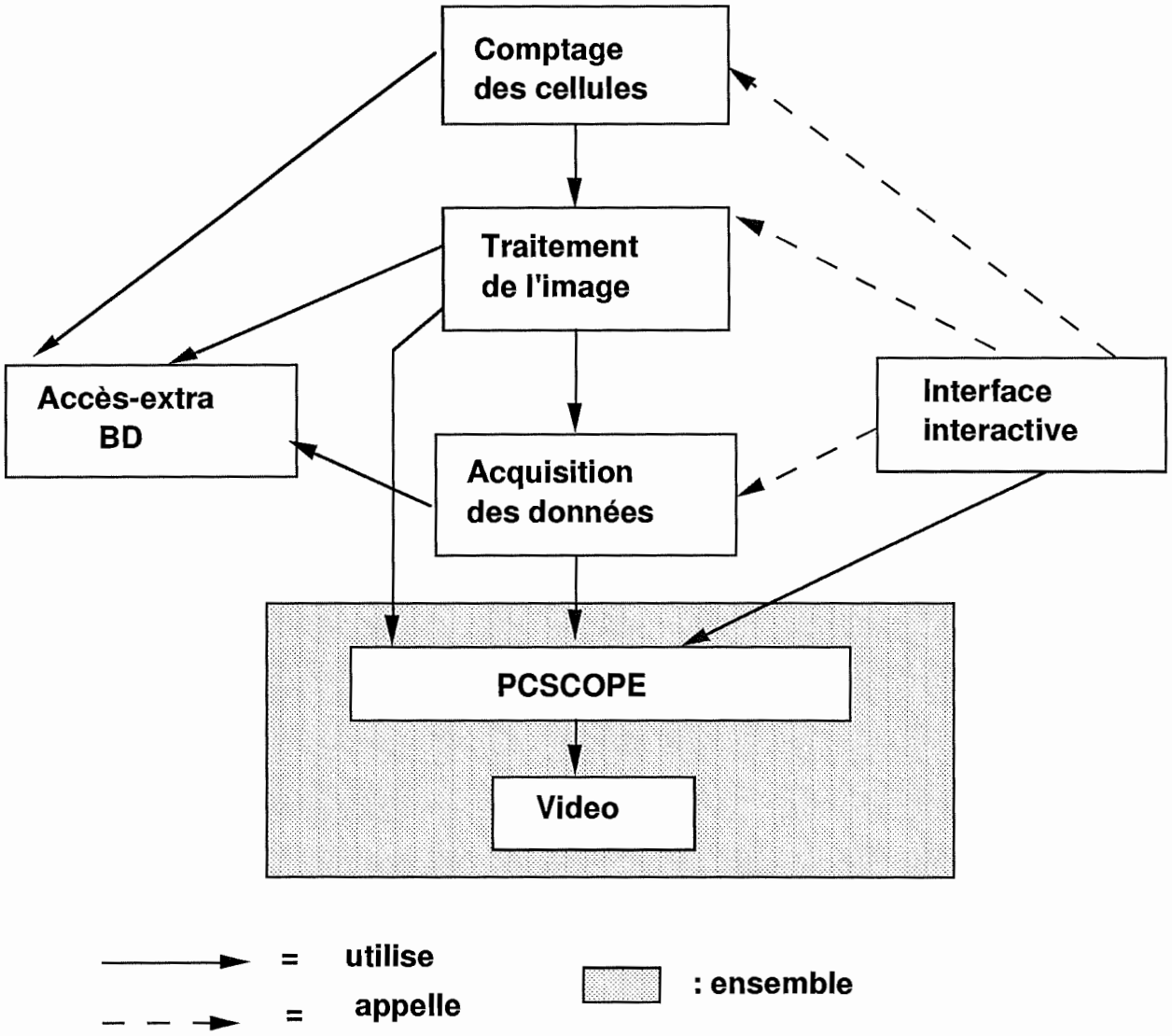

Figure 3-3 L'architecture logicielle du système

Les trois premiers modules sont respectivement décrits de manière plus détaillée dans les chapitres suivants. Tous les autres modules sont brièvement mentionnés dans le chapitre 6.

La liste et les spécifications des différentes primitives ou procédures offertes par ces modules sont reprises en annexe, sauf celles du module **PCSCOPE** qui est un logiciel commercialisé. Pour plus de détails, nous conseillons au lecteur de consulter la référence [PCSC 87].

Chapitre 4 :

Le traitement des données

4.1. Introduction

Ce chapitre présente les deux parties du système qui traitent les données : Acquisition des données et Traitement d'images. Dans la mesure où ces deux parties conditionnent l'ensemble de l'analyse, elles sont très importantes. Il est donc nécessaire de définir les caractéristiques optimales de ces parties compte tenu des conditions matérielles. L'analyse des informations symboliques permet de compléter les critères et fournit, de plus, des informations utiles à la mise en oeuvre de l'étape du comptage de cellules.

4.2. Acquisition des données

L'acquisition des données concerne d'une part la digitalisation de l'image des cellules de l'échantillon donné, et d'autre part la mise en mémoire des différentes informations symboliques annexes nécessaires pour la partie ultérieure du traitement (par exemple, l'identificateur de la boîte, le champ de cellules, la date de l'acquisition, la nature de l'échantillon).

4.2.1 Digitalisation d'images

Pour qu'une image soit traitable par l'ordinateur, elle doit tout d'abord être digitalisée. Trois types de matériel permettent de réaliser la digitalisation d'images [Vinc 87]: les densitomètres; les caméras vidéo et les caméras CCD. C'est ce dernier système que nous avons utilisé (voir Figure 2-1).

Pour la prise des images de cellules, une caméra est montée sur un microscope. Le contrôleur de la caméra est un microprocesseur, qui traite les données et gère les communications entre la caméra et l'ordinateur.

Le problème principal de la digitalisation d'une image réside dans la préservation de la qualité de cette image. Évidemment, il convient tout d'abord d'obtenir une image de bonne qualité, et par conséquent de se placer dans des conditions optimales compte tenu du matériel.

Nous avons procédé comme suit: nous avons

- allumé les alimentations électriques du microscope et une alimentation (ISE52) pour la caméra IVC500 (choix d'une source de lumière assez stable),

- allumé le moniteur TV noir et blanc 36 cm,

- déposé l'échantillon de cellules sous le microscope,

- mis au point le microscope.

Quand l'image est bien nette sur l'écran du moniteur, le signal d'enregistrement est envoyé à l'ordinateur via le clavier. Il est possible de manière interactive de saisir successivement une série d'images.

4.2.2. Acquisition symbolique

Pour compléter le dossier de chaque champ de cellules, un certain nombre d'informations supplémentaires doivent être saisies par le système en même temps que l'image. Les informations les plus importantes sont les suivantes:

- nom de l'opérateur,

- nature de l'échantillon (densité de cellules faible ou importante);

- date de l'acquisition;

- identification de la boîte de Pétri;

- coordonnées du champ, c'est-à-dire la ligne et la colonne dans la boîte de Pétri;

Toutes ces informations sont stockées dans la base de données, et sont accessibles par le système lors de l'étape du traitement de l'image et de la pose du diagnostic. La Figure 4 - 1 montre le processus de l'acquisition de l'image.

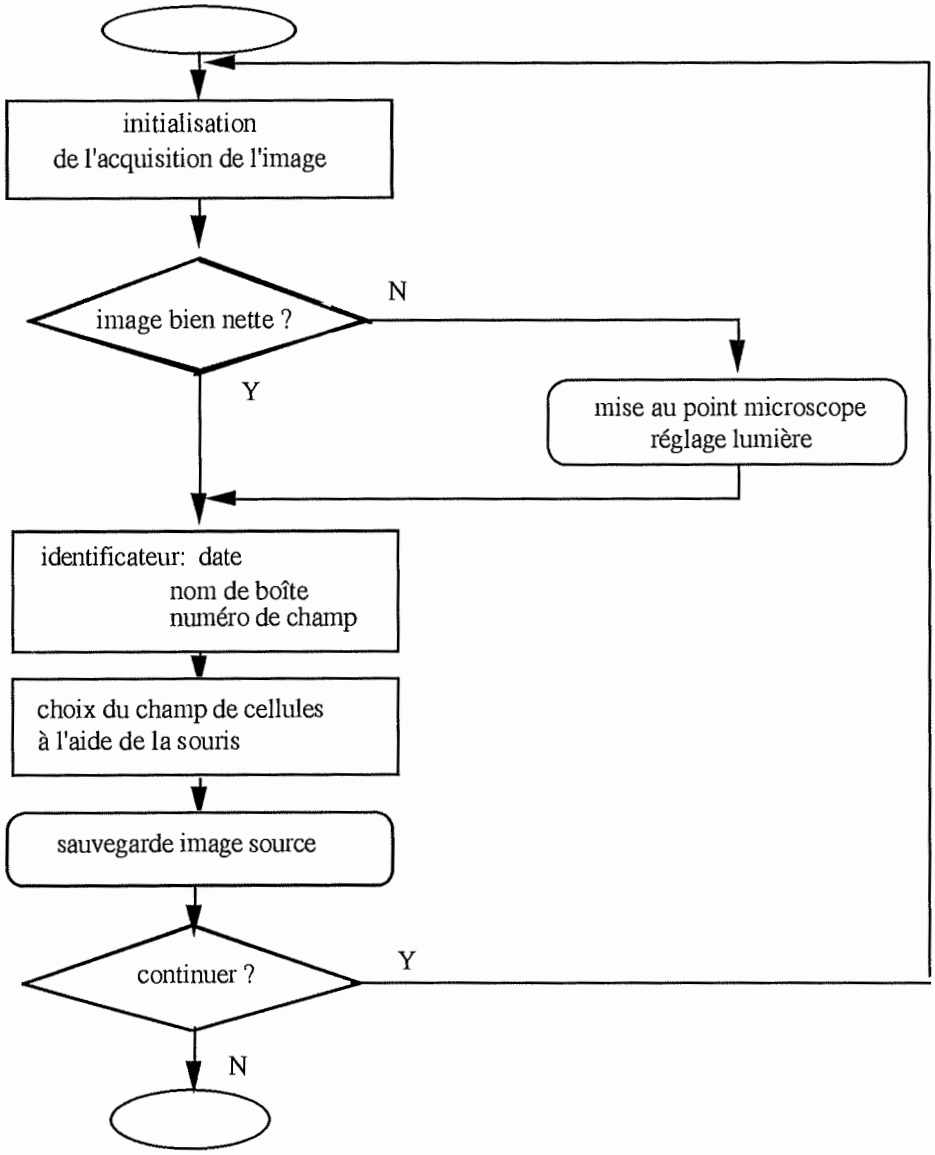

Figure 4-1 Diagramme de l'acquisition des images.

4.3. Traitement de l'image

Le traitement de l'image concerne la manipulation des images originales digitalisées (voir Figure1-2) et leur transformation en d'autres formes. C'est donc une étape préliminaire pour pouvoir faire de l'analyse quantitative sur l'image.

Dans cette section seront exposés les grands principes du traitement d'images et les techniques utilisées dans notre travail.

En examinant l'image originale des cellules, nous notons qu'analyser et quantifier directement cette image par l'ordinateur présentent certaines difficultés.

Dans notre cas, les difficultés sont les suivantes:

- les cellules et le fond se confondent lorsque l'image digitalisée est très obscure (voir Figure 1-2).

- lors de l'acquisition d'une image, les différents bruits sont inévitablement saisis en même temps que l'information significative.

- de plus, la forme des cellules change constamment au cours de la culture (voir la section 5).

Après avoir analysé les images originales, notons que pour réaliser notre objectif d'analyse quantitative des cellules - détecter et compter le nombre de cellules dans un champ de la boîte -, il suffit de s'intéresser à l'information sur le contour des cellules. Dans une image transformée (image traitée), la forme finale des cellules n'est pas importante si la transformation a gardé les informations caractéristiques des cellules originales. Si nous pouvons précisément prédire les contours des cellules, le comptage de celles-ci ne posera pas de problème.

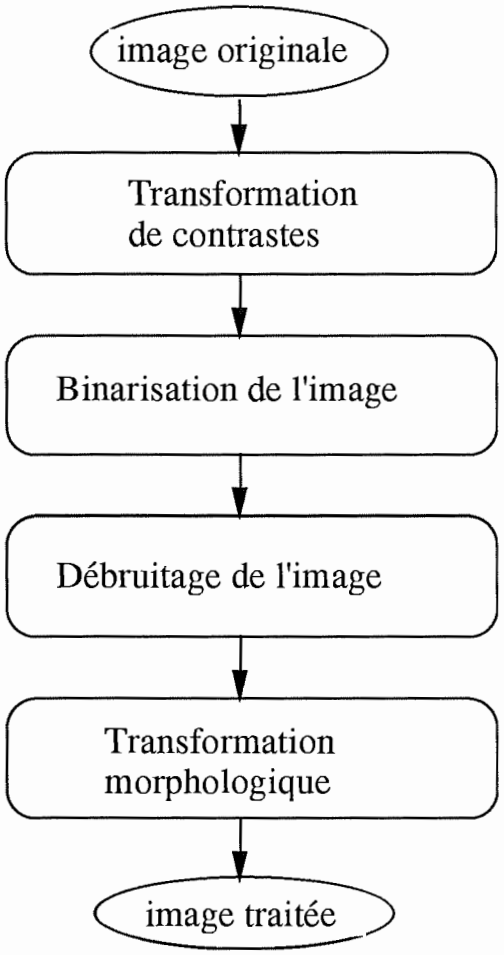

Figure 4-2. Schéma du traitement d'images

Le fait qu'il n'existe aucune méthode générale de traitement de l'image et la spécificité du problème nous ont conduit à adopter une démarche personnelle. Celle-ci a visé à l'obtention d'une précision satisfaisante dans la quantification des cellules (voir Figure 4-2). Les sous-sections suivantes sont consacrées à décrire les principes et les résultats de chacune de ces étapes.

4.3.1. Notions préliminaires

Quand le traitement d'images devient une branche scientifique, il y a beaucoup de termes qui sont définis dans la littérature. Pour que nous puissions mieux expliquer le principe du traitement d'images dans ce chapitre, nous pensons qu'il est utile de rappeler d'abord quelques notions utilisées dans notre travail.

convolution:

La convolution par un noyau (ou masque) permet de réaliser différents traitements d'images. Pour des valeurs bien déterminées de masques, elle permet en particulier d'extraire les contours des objets dans une image.

Principe: Chaque pixel est mis en corrélation avec ses voisins. Le niveau de gris d'un pixel donné est exprimé par une fonction mathématique qui remplace le niveau de gris du pixel par une combinaison des valeurs prédéfinies des niveaux de gris de ses voisins.

Par exemple, pour chaque pixel P_{ij} (i = colonne et j = ligne) dans une image, un noyau de la convolution est défini sur le pixel P_{ij}. Chaque pixel masqué par le noyau est multiplié par la matrice des coefficients exprimant la contribution de ses voisins. P_{ij} devient la somme de ces multiplications.

Dans une voisinage 3 x 3, les pixels autour de P_{ij} sont indexés comme suit:

$P_{i-1,j-1}$	$P_{i,j-1}$	$P_{i+1,j-1}$
$P_{i-1,j}$	$P_{i,j}$	$P_{i+1,j}$
$P_{i-1,j+1}$	$P_{i,j+1}$	$P_{i+1,j+1}$

Si le noyau de convolution est défini :

$K_{i-1,j-1}$	$K_{i,j-1}$	$K_{i+1,j-1}$
$K_{i-1,j}$	$K_{i,j}$	$K_{i+1,j}$
$K_{i-1,j+1}$	$K_{i,j+1}$	$K_{i+1,j+1}$

Le pixel P_{ij} reçoit la valeur $\frac{1}{N}\sum_{x,y} K_{i-x,j-y} * P_{i-x,j-y}$

où **x, y** appartiennent à **{-1, 0, 1}. N** est un facteur de normalisation, plus souvent, $N = \sum_{x,y} K_{i-x,j-y}$.

Nous notons que la convolution peut être effectuée avec des masques de tailles différentes pour réaliser les différents filtrages:

3*1, 3*2, 3*3, 4*4, 5*5, etc.

Le logiciel **PCSCOPE** n'offre que les trois premières possibilités. Ce qui est suffisant pour notre application.

Les différents filtres de convolution peuvent être utilisés dans les cas suivants:

- éliminer les effets parasites (bruits, informations inutiles),
- simplifier l'image (suppression des détails peu visibles),
- améliorer la qualité visuelle,
- faire ressortir les linéarités.

Fonctions morphologiques:

PCSCOPE offre aussi quelques fonctions utiles pour le traitement d'images. Ces fonctions sont orientées-objet et s'appellent "fonctions morphologiques". Ces fonctions sont particulièrement utiles pour extraire et changer la structure des objets concernés dans l'image. Les effets plus importants qu'elles donnent sont, notamment, la réduction des frontières, le remplissage des trous, et le lissage des contours des objets.

4.3.2 Transformation de contrastes

Afin de résoudre la première difficulté que nous avons mentionnée précédemment à savoir: lorsque les cellules et leur environnement se confondent, nous avons commencé par augmenter le contraste entre cellules et fond et par supprimer les détails peu visibles dans l'image.

Il y a plusieurs techniques de filtrage qui peuvent être utilisées, à savoir: **Gradient, Laplacian, Robert, Sobel** [Prat 78] [Opti 88]. Ces filtres sont souvent appelés **"passe-haut"** et ont pour effet de tracer et renforcer le contour des cellules et d'éliminer le bruit de fond.

Après différents essais, nous sommes arrivés à la conclusion que le filtre Sobel convenait le mieux dans notre cas.

En effet, le filtre **Sobel** assigne à chaque pixel la valeur maximale de son gradient horizontal **(X)**, et vertical **(Y)** par les opérateurs suivants :

Sobel en X: $Gx = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$

Sobel en Y : $Gy = \begin{matrix} -1 & 0 & 1 \\ 2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$

Sobel en X et Y : $G = |Gx| + |Gy|$

Le résultat de l'application du filtre **Sobel** à l'image originale est montré à la Figure 4-3:

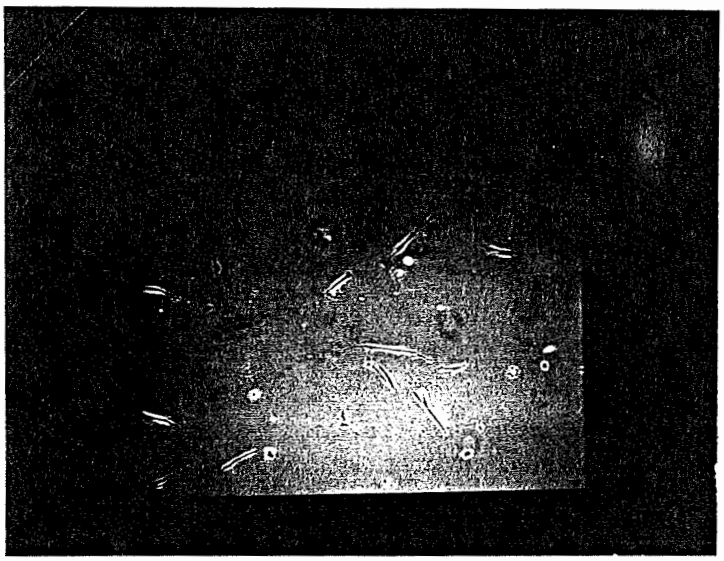

Figure 4-3.a) image avant l'opération Sobel

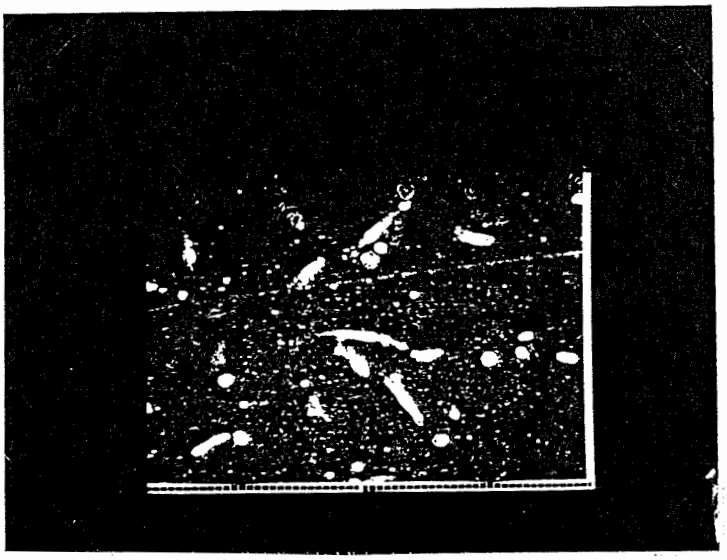

Figure 4-3.b) image après l'opération Sobel

4.3.3. Binarisation de l'image

Etant donné le but de notre travail, il est important que nous puissions détecter précisément le contour des cellules dans une image microscopique. Nous ne nous intéressons donc qu'aux informations sur les contours. Dans la section 4.2.1, nous avons déjà montré que les contours peuvent être renforcés après avoir utilisé les filtres "passe-haut". Le problème est le risque d'avoir les contours des bruits renforcés aussi bien que ceux des cellules. Dans ce contexte, nous sommes conscients que dans l'image, il existe une variance des niveaux de gris parmi le background, les cellules et les bruits. Cette variance n'a pas d'importance, si on peut extraire les cellules du fond. Afin de réaliser cela nous devons recourir à une étape de traitement appelée: "binarisation" ou reéchantillonnage des niveaux de gris, de manière plus générale.

La binarisation de l'image consiste à rechercher un seuil qui sépare l'image en deux régions significatives caractérisées dans l'espace des niveaux de gris. C'est-à-dire:

Si g(i, j) <= S alors g(i, j) = 0;

Si g(i, j) > S alors g(i, j) = 255;

où:

S: Seuil

g(i, j) est la valeur de niveau de gris au point (i, j).

La recherche du seuil de binarisation est facilitée par l'étude des propriétés statistiques de l'image, et en particulier de l'histogramme des niveaux de gris des cellules.

Un histogramme d'une image représente la distribution quantitative des niveaux de gris dans l'image. Dans notre cas, nous utilisons un histogramme qui s'appelle l'histogramme linéaire(voir figure 4-4). Il donne une estimation de la densité de probabilité d'intensité lumineuse dans l'image.

Dans la Figure 4-4, on voit que l'histogramme ne montre qu'un seul pic. Ce pic représente la grande partie du fond qui est éclairé par la lumière. Juste après ce pic, entre les repères que nous avons marqués sur la Figure 4-4, on ne voit qu'un petit nombre de

pixels. Ce sont des informations importantes concernant les cellules. Nous nous y intéressons particulièrement.

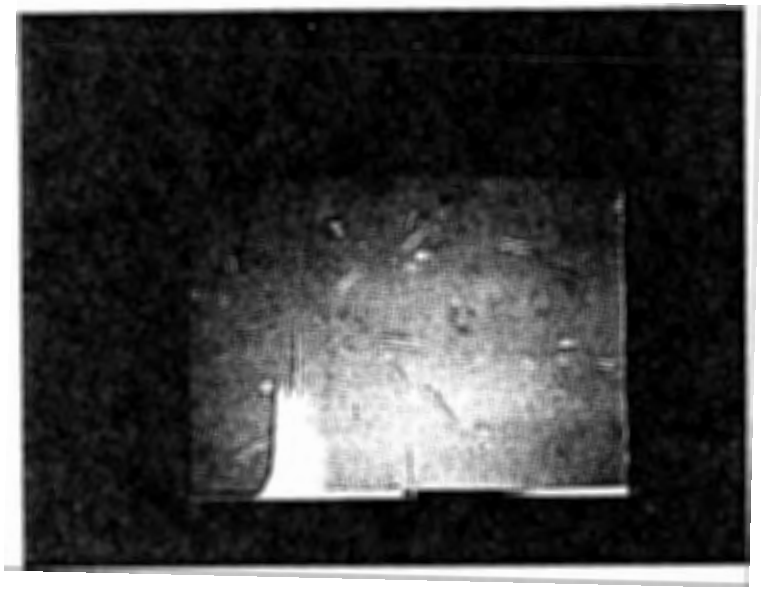

Figure 4-4 L'histogramme linéaire

Nous avons ainsi choisi seuil S = 127 pour la Figure 4-3 a). Après le traitement de binarisation, les cellules sont séparées du fond. Le fond devient homogène et certains bruits sont aussi éliminés (voir le résultat dans la Figure 4-5).

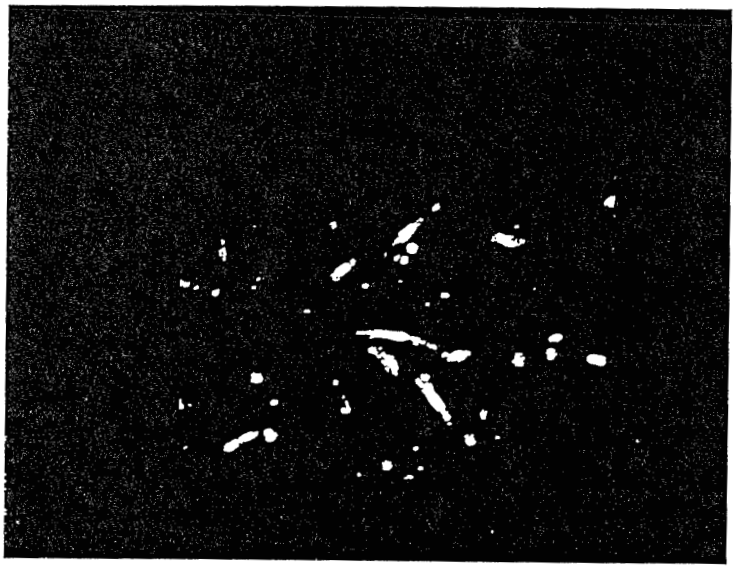

Figure 4-5 Binarisation de l'image

4.3.4. Débruitage de l'image

Dans une image originale d'un champ de cellules, il y a, en principe, trois types de bruits:

1) les taches brillantes qui proviennent des défauts dans le plastique du fond de la boîte de culture (voir Figure 4-3 a).

2) les poussières, ou déchets cellulaires (les tout petits points de couleur foncée).

3) les parasites de type neige (les tout petits points blancs).

Le premier type de bruit est difficile à éliminer par le filtrage de convolution. Dans cette étape-ci, on le traite comme une cellule. Lorsque la surface du bruit de ce type est très petite par rapport à celle d'une cellule, nous pouvons l'éliminer, lors de l'étape de comptage en calculant le nombre de pixels sur le contour d'une cellule (voir la section 5).

Le deuxième type de bruit est éliminé par le filtre Sobel (voir Figure 4-3), parce que le filtrage Sobel assigne à chaque pixel une valeur qui dépend uniquement du niveau de gris de ses voisins, les petites poussières sombres présentes dans l'image sont absorbées par le fond.

D'autre part, certains bruits dus aux poussières peuvent être éliminés par la binarisation de l'image, parce que si les niveaux de gris des bruits sont inférieurs à la valeur du seuil que l'on a choisi, ces bruits sont considérés comme le fond, et les niveaux de gris sont mis à zéro.

Maintenant, il faut nous attacher à éliminer les bruits parasites de type neige. Le filtre Médian peut être utilisé à cet effet.

Le filtre médian est normalement appelé filtre "non-linéaire" qui donne à chaque pixel un niveau de gris par une fonction de ses voisins. Il consiste en une fenêtre qui se déplace sur les pixels de l'image et englobe un nombre impair de points, le point central étant remplacé par la valeur du point médian de la fenêtre.

Le point médian est défini de manière à ce que (N-1)/2 points aient des valeurs plus petites et (N-1)/2 points aient des valeurs plus grandes que lui. Par exemple, dans la séquence 107,108,180,112,116 le point central 180 sera remplacé par la

valeur 112 qui est la valeur médiane de la séquence ordonnée 107,108,180,112,116.

L'avantage de cette méthode est que si le point 180 provient du bruit, il sera alors avantageusement éliminé, tandis que si dans son voisinage on trouve d'autres points semblables, le point médian sera le même et la valeur sera conservée. Il s'ensuit que le contour de la cellule ne sera pas estompé.

La Figure 4-6 montre le résultat du filtre Médian avec la fenêtre de la taille3 x 3.

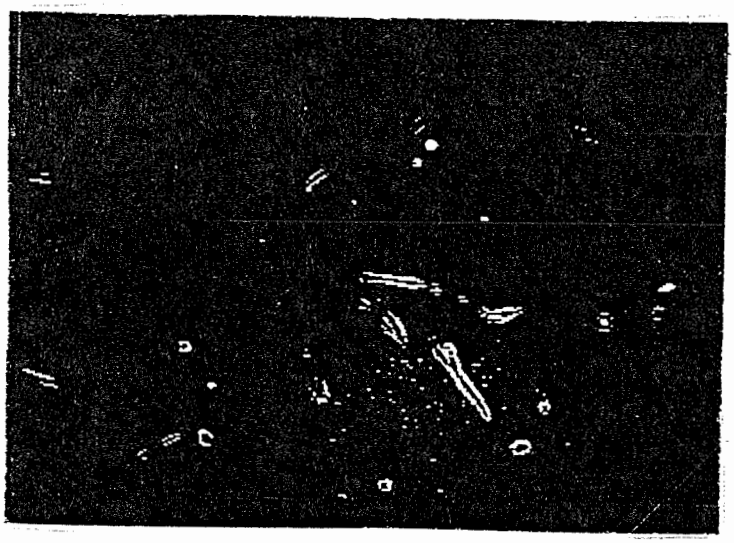

Figure 4-6 a) Image avant filtrage médian

Figure 4-6. b) image après filtrage médian

4.3.5. Transformation morphologique

Dans tous les traitements d'images que nous avons décrits dans les sections précédentes, nous avons considéré l'amélioration de l'image dont l'objectif principal est d'extraire les informations des objets intéressants du fond, de les rendre plus clairs, et d'éliminer les bruits.

Les objets intéressants pour nous sont des cellules. Maintenant, elles apparaissent avec plus d'évidence en regardant la Figure 4-7.

Malheureusement, il est encore très difficile pour l'ordinateur de les reconnaître et de les compter automatiquement, parce que, d'une part, les contours des cellules ne sont pas toujours fermés, d'autre part, certaines cellules sont superposées.

Au vu de ces problèmes, nous utilisons les deux fonctions morphologiques: "**Erosion**" et "**Dilatation**".

La fonction **Erosion** peut réduire le degré de luminosité de chaque pixel qui est entouré par des pixels de luminosité moins forte, c'est-à-dire: soit P_0 le pixel en question et P_i les pixels voisins de P_0. L'application de cette fonction donne

$$P_0 = \mathbf{Min}\ (P_i)$$

La fonction dilatation est l'inverse de l'Érosion, i.e.

$$P_0 = \mathbf{Max}\ (P_i)$$

En effet, ces deux fonctions sont utilisées le plus souvent l'une après l'autre. Cela dépend du problème envisagé. Dans notre cas, nous employons d'abord, **Dilatation** et ensuite **Erosion**.

L'effet important de ces deux fonctions est le lissage et remplissage des contours (voir la Figure 4-7).

Après cette opération (**Dilatation suivie de Erosion**), on voit que l' intérieur des cellules est rempli, les contours sont fermés et certaines cellules superposées sont séparées.

Figure 4-7 Image après Dilatation suivie de Erosion

Chapitre 5 :

Comptage des cellules

5.1. Introduction

Dans le chapitre 4, nous avons exposé une série de traitements d'images. Tous ces traitements ont pour but de permettre un comptage de cellules précis et automatique. Il est clair que cette étape de comptage est une étape critique dans le fonctionnement du système.

Nous allons voir dans ce chapitre l'algorithme de la détection des cellules dans un champ d'une boîte de Pétri, et l'algorithme de la détection du contour des cellules. Nous verrons que les algorithmes ici peuvent être étendus au débruitage et à d'autres analyses quantitatives.

5.2 Détection des cellules

Pour rappel, une image digitalisée est un ensemble de pixels. Si l'on ne s'intéresse qu'aux niveaux de gris des points de l'image, il convient d'utiliser la notion suivante **g(x, y)** qui représente le niveau de gris au point (x, y). Dans une image binarisée, les niveaux de gris des pixels n'ont que deux valeurs possibles:

255 (blanc) pour les pixels d'une cellule,

0 (noir) pour les pixels du fond.

Pour déterminer les cellules dans ce genre d'image (voir la **Figure** 5-1), nous utilisons la méthode suivante (étape du balayage):

1. se positionner au point original (0,0) de l'image et initialiser le compteur de cellules à 0;

2. rechercher les pixels d'intensité égale à 255 dans les directions horizontales (de gauche à droite) et verticales (de haut en bas);

3. tant que le point courant du balayage décrit ci-dessus est le premier point d'intensité 255 qui n'a jamais été exploité précédemment, on définit ce point comme le point de départ pour détecter le contour de la cellule (voir la section suivante);

4. tant que la procédure de la détection du contour se termine, augmenter le compteur de cellules de un, et retourner à l'étape 2.

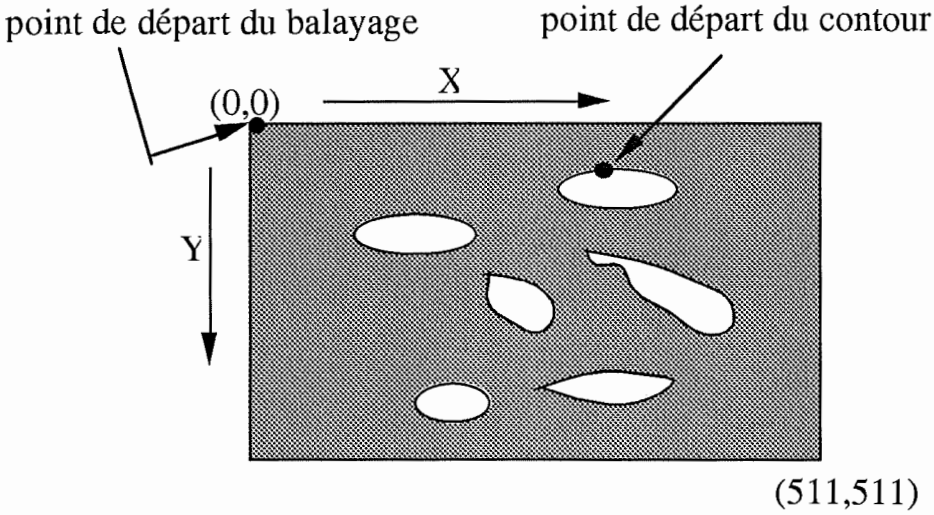

Figure 5-1 Détection des cellules (balayage)

5.3. Détection du contour

Dans la section précédente, nous avons montré le balayage pour rechercher des cellules dans une image concernée. Lorsque le premier point (x_0, y_0, $gris_0$) du contour à déterminer a été trouvé dans l'étape du balayage, il suffit, à partir de ce point, de rechercher le point voisin (voir Figure 5-2) d'intensité égale à 255 qui se trouve sur le contour. La boucle continue jusqu'à ce que l'on retourne au point de départ.

Parce que les contours dans une image ne sont pas uniformes et que les contours eux-mêmes ne sont pas lisses, leur détection est assez difficile à mettre en oeuvre. Nous n'avons pas trouvé dans la littérature un algorithme de détection de contour décrit en détails et de manière très satisfaisante. Nous voudrions donc dans cette section décrire davantage et de façon plus compréhensible l'algorithme que nous avons élaboré. Nous donnons non seulement l'algorithme décrit en pseudo-langage, mais aussi le diagramme de cet algorithme (voir la figure 5-6).

Pour déterminer les pixels du contour, on doit d'abord déterminer la direction de la détection. La figure 5-3 montre les 8 directions possibles de détection.

O3(X0-1,Y0-1)	O2(X0,Y0-1)	O1(X0+1,Y0-1)
O4(X0-1, Y0)	O (X0, Y0)	O0(X0+1, Y0)
O5(X0-1,Y0+1)	O6(X0, Y0+1)	O7(X0+1,Y0+1)

Figure 5-2. Les points voisins du point O(x0,y0).

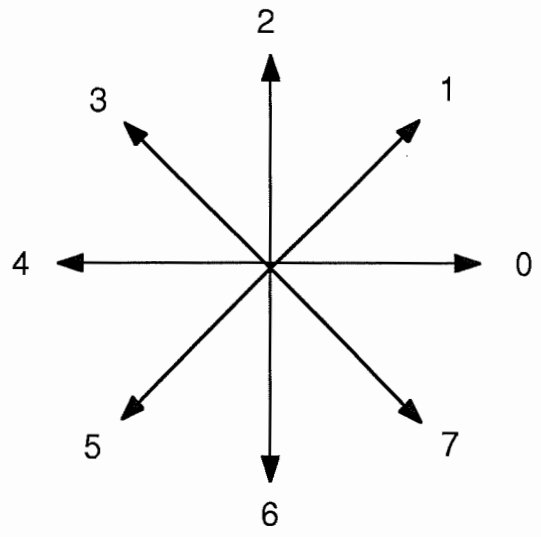

Figure 5-3. Les directions de la détection du contour

Définition des variables :

{ P }: l'ensemble des pixels (ou points) du contour d'une cellule;

C : le point courant à déterminer;

B : le point voisin du point C et le point suivant à déterminer;

D : la direction de détection du contour codée sur un entier de 0 à 7, (voir la Figure 5-3);

Pd : le point voisin du point P dans la direction D, par exemple, C1 est un point voisin du point C dans la direction 1;

First: la variable de repère. Quand l'algorithme commence, une valeur 1 lui est assignée;

Found: la variable de repère. Quand l'algorithme trouve le point suivant du contour, une valeur 1 lui est assignée,

Compp: le compteur des pixels du contour d'une cellule. L'avantage de créer un tel compteur est double. D'une part, on l'utilise pour éliminer les bruits de l'image, d'autre part, pour segmenter les deux cellules qui se touchent (voir section 5.3).

L'algorithme de détection du contour

0. Sélectionner un point de départ du contour, A. Il est évident que le point voisin A4 n'est pas un point de la cellule. Ceci est garanti par l'algorithme de balayage.

1. Assigner - A à C (C = A);
- 6 à D (D = 6);
- 1 à First (First = 1);
- 0 à Compp (Compp = 0).

2. Tant que la valeur de la variable C n'est pas égale à celle de A ou que la variable First est égale à 1 ((C != A) || (First

==1)), procéder aux étapes 3-10.

3. Assigner 0 à Found (Found = 0).

4. Tant que la variable Found est égale à 0 (Found == 0), procéder aux étapes 5 - 9.

5. Assigner C_{D-1} (qui est le point voisin dans la direction (D-1) du point C) à B.

Si B est le point du contour, procéder à l'étape 6,
Sinon procéder à l'étape 7.

6. Assigner - B à C (C = B);
- (D - 2) à D (D = D - 2);
- 1 à Found (Found = 1);
- 1 à Compp (Compp++),

et passer à l'étape 10.

7. Assigner C_D à B.

Si B est le point du contour,
assigner B à C (C = B); 1 à Found (Found = 1); et incrémenter à Compp (Compp++), et passer à l'étape 10,
sinon procéder à l'étape 8.

8. Assigner C_{D+1} à B (B = C_{D+1}).

Si B est le point du contour,
assigner B à C (C = B), 1 à Found (Found = 1), et 1 à Compp (Compp++), et passer à l'étape 10.
Sinon passer à l'étape 9.

9. Assigner (D+2) à D (D = D+2), retourner à l'étape 4;

10. Assigner 0 à First (First = 0), retourner à l'étape 2.

11. Fin de l'algorithme

Note: - ce qui est écrit entre les parenthèses sont des expressions en langage C. Ceci a pour but d'aider le lecteur à lire le diagramme de l'algorithme (voir la Figure 5-6).

- le calcul du code D de la direction se fait modulo 8.

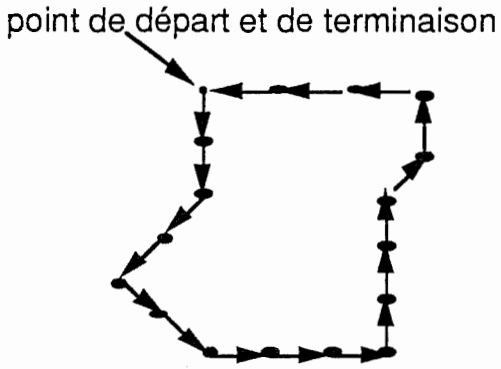

Figure 5-4. L'exemple de la détection du contour

La figure 5-5 montre le résultat après avoir appliqué l'algorithme de la détection des cellules à l'image de la figure 4-7.

Figure 5-5 Image après avoir appliqué l'algorithme de la détection des cellules

Diagramme de l'algorithme

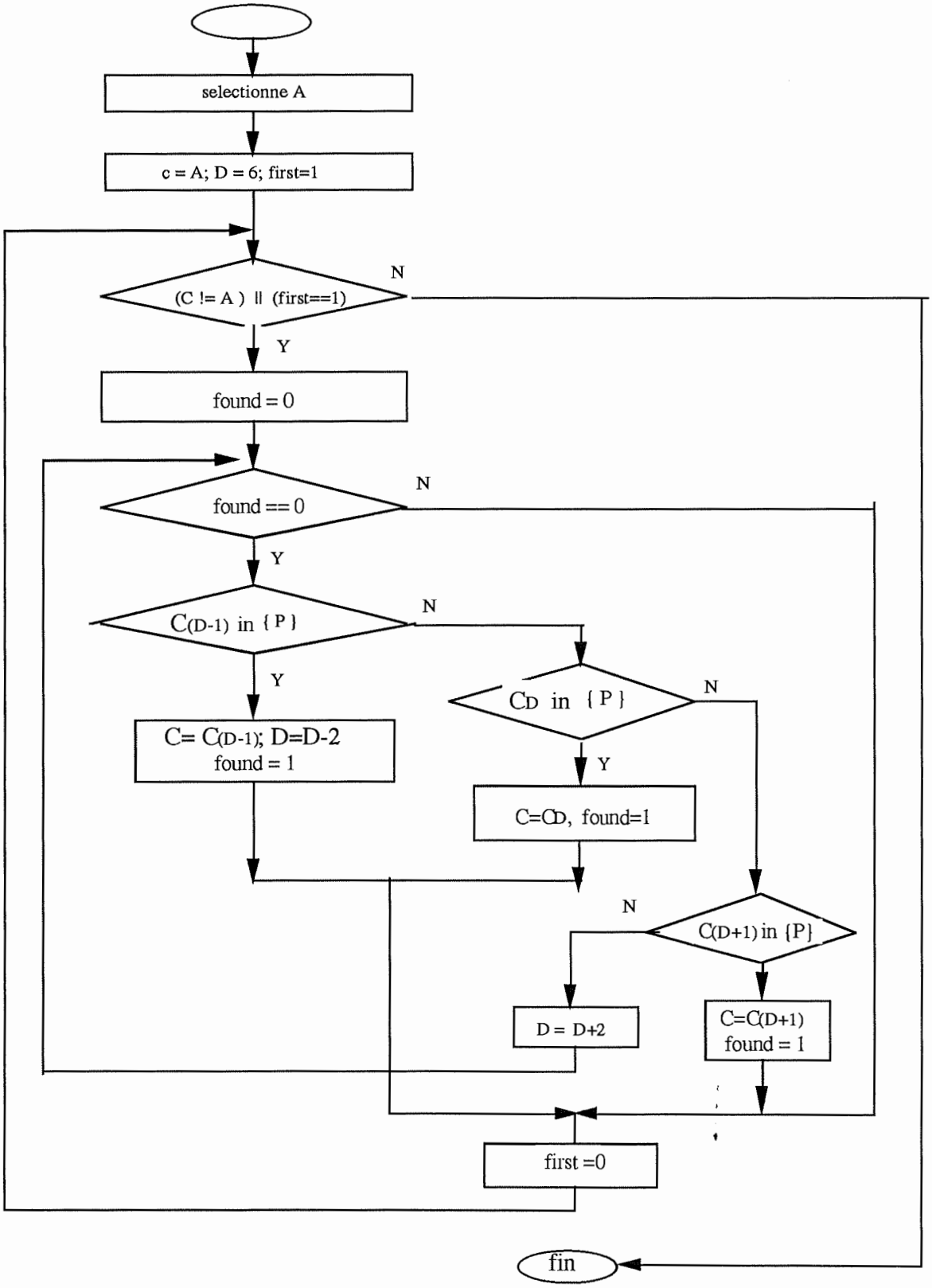

Figure 5-7 Diagramme de la détection du contour

5.4 Extension de l'algorithme de comptage

5.4.1 Débruitage et segmentation

Dans la section 4.2.3, nous avons brièvement mentionné que certains bruits de type tache brillante peuvent être éliminés par le programme de comptage des cellules.

En effet, quand l'algorithme fonctionne pour détecter les pixels sur le contour d'une cellule, le compteur augmente en même temps. Quand l'algorithme de détection du contour de la cellule se termine, le compteur a une valeur qui représente le nombre de pixels dans le contour de cette cellule. Si cette valeur est très petite, on peut considérer que la "cellule" que l'algorithme vient de détecter n'était pas une vraie cellule, mais une tache. On la rejette sans incrémenter le compteur qui représente le nombre de cellules dans une image.

Avec la même idée, si le compteur contient une valeur trop élevée, on peut considérer que la "cellule" que l'algorithme vient de détecter est un ensemble de deux ou trois cellules qui se chevauchent. Dans ce cas-là, le compteur de cellules peut augmenter de 2 ou 3 au lieu d'un.

Bien entendu, toutes ces opérations (débruitage et segmentation) doivent se faire selon des critères prédéfinis. Par exemple, si la valeur du compteur est inférieure à 20, ce n'est pas une cellule; et si la valeur est plus grande que 80, ce sont des cellules imbriquées.

La figure 5-7 montre que certains petits points dans la figure 5-5 sont éliminés aprés avoir appliqué le programme de la détection des cellules. A savoir, le critère d'élimination est 20.

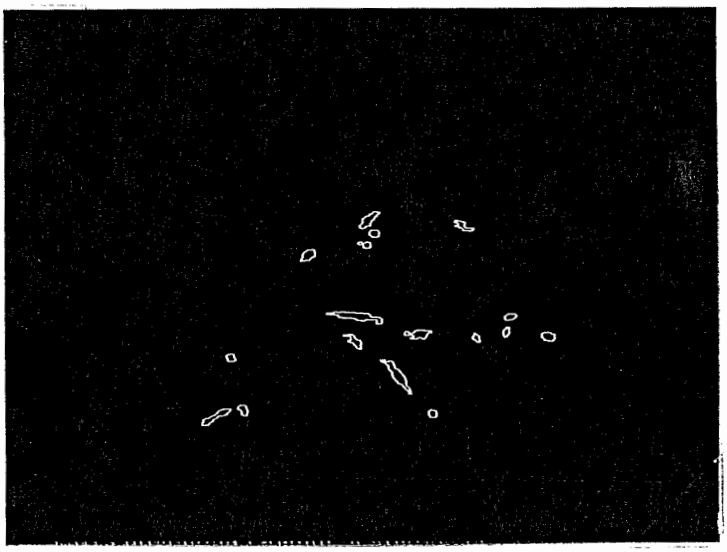

Figure 5-8 Elimination des bruits en fonction du nombre des pixels du contour de la cellule

5.4.2 Autres analyses quantitatives

Le compteur peut aussi être utilisé pour calculer les superficies des différentes cellules. Ce genre de mesure peut être très intéressant pour les biochimistes qui veulent analyser les cellules de différents types morphologiques. Notons que notre système ne convient pas pour faire une analyse morphologique des cellules, parce que les formes originales des cellules sont déjà transformées. Si on veut faire ce genre d'analyse avec l'ordinateur, la transformation de l'image doit garder les caractéristiques des formes originales des objets intéressants. Cela mériterait une recherche plus approfondie (voir la section 7.4. Perspectives de développement).

Chapitre 6 :

Implémentation des fonctionnalités du système

6.1 Introduction

Dans les chapitres précédents, nous avons, à partir d'un aspect théorique, montré les deux parties principales des fonctionnalités du système: le traitement d'images et le comptage des cellules. Dans ce chapitre, nous allons, à partir de l'aspect pratique, montrer la réalisation des fonctionnalités du système telles qu'elles ont été décrites dans le chapitre 3.

Les difficultés qui surgissent lors de l'implémentation sont de deux types. Il s'agit d'abord de difficultés liées à la recherche d'un seuil pour la binarisation des images. Pour que le traitement d'images soit automatique, on ne peut pas sélectionner manuellement un seuil chaque fois que l'on envisage de traiter une nouvelle image. Il s'agit ensuite de difficultés liées à la détection des contours des cellules. Lorsque les contours des cellules sont très variés et certains très compliqués, on doit faire attention pour concevoir l'algorithme de détection.

Nous montrons dans les sections 6.5. et 6.6 comment nous avons surmonté ces difficultés.

Les programmes réalisés dans ce mémoire sont écrits en langage C (Microsoft Quick C). Nous avons choisi le langage C pour les raisons suivantes: premièrement, les fonctions de PCSCOPE sont aussi écrites en langage C. Donc, il nous permet d'avoir un meilleur

usage de PCSCOPE et une meilleure compatibilité entre notre système et PCSCOPE. Deuxièmement, il est bien connu que l'exécution des programmes en langage C est rapide, puisque le langage C permet de gérer efficacement des structures de données qui optimisent la vitesse d'exécution.

En se référant à la Figure 3-3, chaque module est implémenté sous forme d'une unité de compilation séparée. Chacune des procédures offertes par les modules réalise une des fonctionnalités proposées par le système. Les procédures offertes par le module interface interactive sont directement appelées lors de leur sélection par l'utilisateur, dans le menu principal.

6.2. Interface (module d'interface interactive)

Le rôle de l'interface est de permettre à l'utilisateur d'effectuer une ou plusieurs tâches à l'aide du système informatique mis à sa disposition.

Lorsque les utilisateurs de notre système sont des biochimistes, le système doit être "transparent", c'est-à-dire cacher la complexité de l'aspect informatique, surtout le traitement d'image.

Du point de vue des utilisateurs du système, nous pensons que deux services essentiels doivent être offerts par le système. Le premier est le service d'acquisition des images de cellules et le deuxième est le comptage des cellules (voir les sections suivantes).

Les deux services offerts par le système sont proposés à travers le menu principal à l'écran. La structure du menu est reprise à la Figure 6-1.

Menu principal:

- **Acquisition de l'image individuelle** **(1)**
- **Acquisition interactive d'images** **(2)**
- **Comptage des cellules de l'image individuelle** **(3)**
- **Comptage automatique de cellules d'images** **(4)**
- **Quitter** **(5)**

Figure 6-1 Le menu principal du système

Ce menu principal comprend quatre sous-menus. Chacun de ces sous-menus comporte une série de dialogues entre le système et l'utilisateur. Les champs d'entrée de données, messages de service ou d'erreur apparaissent sur l'écran de l'ordinateur. L'instrument de dialogue est le clavier, la souris étant réservée pour les manipulations sur la mémoire d'image lors de l'acquisition des images.

Les messages seront relativement nombreux pour informer les utilisateurs sur la manière d'entamer le processus.

6.3 Organisation des données (module d'accès-extra BD)

L'analyse des différentes fonctions offertes par le système met en évidence que l'on doit créer un module d'accès et d'extraction des informations manipulées par le système. Les informations importantes sont les suivantes:

- l'information "bruit" de départ qui est l'image digitalisée d'un champ de cellules dans une boîte de Pétri.

- les informations symboliques relatives aux images acquises.

- les informations des contours des cellules lors du comptage des cellules.

- les points maximaux et minimaux relatifs aux cellules.

Les informations sont organisées sous forme de fichiers qui sont stockés sur le disque et appelés la base de données (BD). A chacun de ces quatre types d'information correspond un type de fichier qui lui est propre:

- **Fichier de type** **.fim*. Ce type de fichier contient l'information relative à la taille de l'image ainsi que l'intensité codée sur un byte de chaque point de l'image. La gestion d'un tel fichier se fait entièrement à l'aide des fonctions offertes par PCSCOPE. (Pour de plus amples informations, consulter le manuel d'utilisation PCSCOPE).

- **Fichier de type** *symb.p*. Ce type de fichier contient les informations symboliques relatives à l'image digitalisée d'un champ de cellules lors de l'acquisition d'images. La structure d'un tel enregistrement est la suivante.

symb.p :

Symbol, Symbol, ... Symbol.

Où: Symbol

Byte	Type	Contenu
1 à 8	string	nom de fichier.fim
9 à 10	int	ligne (location d'un champ de cellules en x)
11 à 12	int	colonne (location d'un champ de cellules en y)
13 à 21	string	date de l'acquisition
21 à 31	string	nom de l'opérateur
32 à 33	int	type de cellules
34 à 35	int	nombre de cellules

Figure 6-2 La structure du fichier symb.p

Notons que le dernier item du fichier est toujours égal à zéro au moment de l'acquisition de l'image. On lui assigne une valeur seulement lorsque le comptage des cellules de l'image a été fait.

- **Fichier de type *cell.p*** Ce type de fichier contient les informations concernant tous les pixels du contour d'une cellule détectée. Ce fichier est utile quand on veut retracer les contours des cellules. La structure d'un tel enregistrement est la suivante.

cell.p:

cellule, cellule, ... cellule

Où: cellule

Byte	Type	Contenu
1 à 2	int	ligne (position en x dans une image de cellules)
3 à 4	int	colonne (position en y dans une image de cellules)
5 à 6	int	numéro de la cellule

Figure 6-3 La structure cellule

Notons que le niveau de gris d'un pixel du contour est toujours égal à 255 dans une image binaire. Il n'est pas donc nécessaire de le stocker, ce qui ainsi permet de gagner de la mémoire.

- **Fichier** ***maxmin.p***. Ce type de fichier contient les informations concernant les quatre valeurs du contour d'une cellule, i.e. Max{Xi, Yi} et Min{Xi, Yi}. Où i varie de 1 à P (le nombre de pixels du contour d'une cellule).

En effet, une cellule dans une image peut être considérée comme un rectangle montré dans la figure 6-4. Ce sont des informations utiles, lors du comptage des cellules.

Rappelons que l'algorithme de balayage se fait de haut en bas et de gauche à droite. Chaque fois que l'on rencontre un pixel qui a un niveau de gris égal à 255, on doit s'assurer que ce pixel n'a jamais été exploité par l'algorithme de détection du contour. Si c'est le cas, on passe au point suivant, sinon on appelle l'algorithme de détection du contour à partir de ce pixel-ci. Donc, il est clair que le fichier ***maxmin.P*** sert à représenter les pixels des contours qui ont déjà été exploités.

Ce type de fichier est un ensemble de tableaux de l'ordre 4:

minmax[4], minmax[4], ... minmax[4],

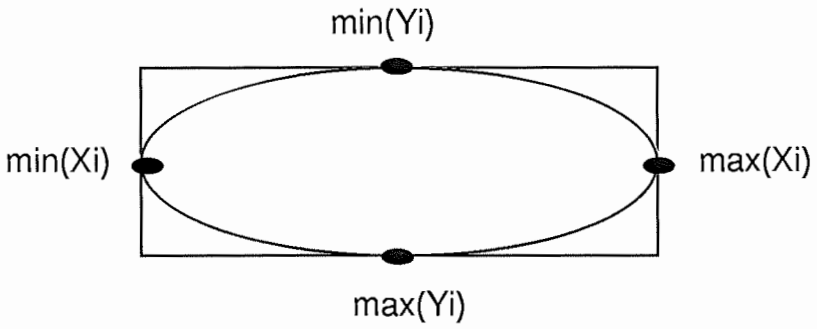

Figure 6-4 Les points maximaux et minimaux d'une cellule

Notons que les quatre types de fichiers décrits ci-dessus constituent des fichiers à accès séquentiel.

- **Chaînage des pixels du contour.** Une structure de données annexes (liste chainée) a été implémentée pour optimiser l'accès aux données et pour avoir une meilleure performance de l'algorithme du comptage des cellules. Ceci a pour but de créer une zone tampon pour stocker les informations concernant le contour d'une cellule pendant la détection du contour. La structure de la liste est la suivante:

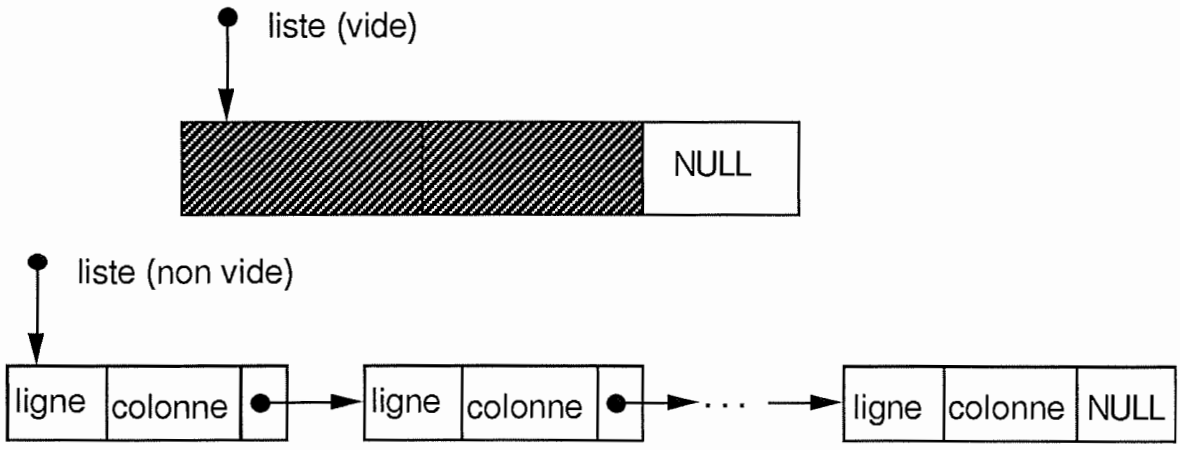

Figure 6-5 Chaînage des pixels du contour

6.4. Acquisition des données (module d'acquisition des données)

En ce qui concerne la fonctionnalité de ce module, nous en avons déjà parlé à la section 4.1. Ici nous voulons simplement rappeler que le module est implémenté en mode interactif, puisque l'utilisateur doit fournir certaines informations concernant l'image à saisir. L'utilisateur peut appeler directement ce module à l'aide du menu principal.

6.5. Automatisation du traitement d'images (module de traitement de l'image)

L'automatisation du traitement d'images dépend de la recherche du seuil de binarisation. Il existe plusieurs techniques pour rechercher ce seuil. Dans le cas idéal, l'histogramme montre une profonde vallée entre deux pics qui représentent l'un les objets et l'autre le fond de l'image. Dans ce cas-ci, le seuil peut être choisi au fond et au milieu de ces vallées [Prew 66] (voir aussi Annexe B).

Malheureusement, beaucoup d'histogrammes d'images réelles ne montrent pas les deux pics, à cause de la faiblesse du contraste entre les objets intéressants et le fond de l'image, et à cause des bruits.

Parce que les images acquises peuvent être très différentes l'une de l'autre, le choix du seuil de binarisation doit être généralisé d'une façon sûre. Pour ce faire, nous avons dû créer une procédure **trouve_seuil()** qui calcule l'histogramme de chaque image avant que l'on procède au traitement de cette image. Après calcul, on trouve le niveau de gris (**gris-pic**) qui correspond au pic de l'histogramme de l'image. Dans notre cas, on sait que cette valeur représente plutôt le fond de l'image. Le seuil doit être choisi juste après la pente à droite du pic. Il suffit donc d'ajouter une constante empirique, à savoir 15, à la valeur du pic pour obtenir le seuil. Donc, **le seuil = 15 + gris-pic**. Dans tous les tests que nous avons faits, les résultats ont été satisfaisants.

6.6 Automatisation du comptage des cellules (module comptage des cellules)

Il n'y avait pas trop de difficultés à réaliser le comptage automatique de cellules sur plusieurs images de cellules se basant sur les informations du fichier ***symb.p***, il nous suffit d'appeler pour chaque image successivement la procédure de détection des cellules (voir la section 5).

La difficulté est de concevoir un bon algorithme de détection des cellules. Pour que la boucle dans l'algorithme de la détection de la cellule ne soit pas une boucle infinie, on doit s'assurer que le point de départ soit aussi un point de sortie de l'algorithme. C'est-à-dire que la détection doit réellement se poursuivre le long du contour de la cellule, bien que le contour de la cellule ne soit pas lisse. N'importe quel raccourcissement de la détection du contour va conduire à une boucle infinie.

Chapitre 7 :

Discussion

On a vu que le traitement d'images digitalisées est important étant donné l'hétérogénéité du champ, des cellules et la présence de contaminants qui peuvent être traités comme des cellules par le procédé informatique.

Il était donc crucial de pouvoir confronter le comptage par le système que nous avons mis au point et le comptage manuel tel qu'il est effectué au laboratoire.

7.1 Comparaison des résultats entre comptage automatique et comptage humain

Soit le résultat d'un comptage à l'oeil nu **oeil** et celui d'un comptage par notre système **ord**. Afin de tester s'il y a une grande différence entre les deux types de comptage, nous avons échantillonné 47 images. Les résultats des mesures sont repris dans le tableau 7-1.

D'après ces résultats, on peut se demander s'il y a une grande différence entre les deux types de comptage.

La réponse est la suivante:

champ.fim	nombre de cellules comptées à		di=ord-oeil	di^2
	oeil	ord		
f1	6	5	-1	1
f2	10	10	0	0
f3	14	12	-2	4
f4	10	13	3	9
f5	6	7	1	1
f6	10	9	-1	1
f7	7	6	-1	1
f8	4	4	0	0
f9	6	6	0	0
f10	8	7	-1	1
f11	10	9	-1	1
f12	10	10	0	0
f13	12	12	0	0
f14	5	4	-1	1
f15	7	6	-1	1
f16	8	8	0	0
f17	6	5	-1	1
f18	10	9	-1	1
f19	5	6	1	1
f20	13	16	3	9
f21	16	14	-2	4
f22	19	19	0	0
f23	14	13	-1	1
f24	25	24	-1	1
f25	22	20	-2	4

Tableau 7-1 a) Les résultats des mesures des deux méthodes de comptage

champ.fim	nombre de cellules comptées à		di=ord-oeil	di^2
	oeil	ord		
f26	13	12	-1	1
f27	14	14	0	0
f28	15	15	0	0
f29	15	12	-3	9
f30	14	14	0	0
f31	13	11	-2	4
f32	8	8	0	0
f33	48	47	-1	1
f34	21	21	0	0
f35	34	35	1	1
f36	48	47	-1	1
f37	18	15	-3	9
f38	36	37	1	1
f39	16	19	3	9
f40	18	21	3	9
f41	35	33	-2	4
f42	36	34	-2	4
f43	49	52	3	9
f44	39	36	-3	9
f45	25	30	5	25
f46	36	40	4	16
f47	39	38	-1	1
total			-8	156

Tableau 7-1 b) Les résultats des mesures des deux méthodes de comptage

Si la qualité des deux sortes de comptage est la même, et si le résultat du comptage à l'oeil nu est consideré correct (100%), alors, la différence entre la paire de données dans le tableau 7-1, est due aux erreurs aléatoires. Cependant, la distribution des erreurs aléatoires peut être considérée comme la distribution normale avec une moyenne(μ). Donc la différence des deux comptages **d = ord - oeil** doit satisfaire aux caractéristiques de la distribution normale, i.e. $N(\mu, \frac{\sigma}{\sqrt{n}})$, c'est-à-dire, la moyenne des différences est:

$$\overline{d_n} = \frac{\sum d_i}{n} \sim N(\mu, \frac{\sigma}{\sqrt{n}}) \qquad < I >$$

Donc, le problème revient à vérifier l'hypothèse concernant la différence des résultats provenant de deux comptages différents:

H_0: $\mu = 0$.

H1: μ <> 0: indique que l'on doit faire un test bilatéral.

Ceci veut dire que si $| \overline{d_n} | > \varepsilon$, alors on rejette H_0. D'après la théorie du test des hypothèses, la probabilité de rejeter H_0 doit être très petite, i.e.:

$$P_r(| \overline{d_n} - \mu_0 | > \varepsilon \mid \mu = 0) = \alpha \quad (0<\alpha<1) \qquad < II >$$

c-à-d:

$$P_r(\varepsilon \leq \overline{d_n} \leq \varepsilon) = \alpha \qquad < III >$$

où α est le niveau de signification du test.

Lorsque la variance σ^2 est inconnue, nous prenons S^2 comme un estimateur de σ^2.

$$S^2 = \frac{\sum (d_i - \overline{d_n})^2}{n - 1} \qquad < IV >$$

i.e.

$$S^2 = \frac{n}{n - 1} (\frac{\sum d_i^2}{n} - \overline{d_n}^2) \qquad < V >$$

Dans ce cas,

$$\frac{\overline{d_n}}{S/\sqrt{n}} \sim t(n-1) \qquad \text{< VI >}$$

où **t(n-1)** est la distribution **t** de student à n-1 degrés de liberté [Feyt 91]. Ce qui fournit la région critique (c-à-d: région de rejet pour l'hypothèse H_0) au niveau α:

$$\frac{|\overline{d_n}|}{S/\sqrt{n}} > t_{\frac{\alpha}{2}}(n-1) \qquad \text{< VII >}$$

Dans notre cas, le tableau 7-1 fournit les valeurs suivantes:

$n = 47;$

$\overline{d_n} = -0.17;$

$S = 1.83$

Alors,

$$\frac{|\overline{d_n}|}{S/\sqrt{n}} = 0.63 \qquad \text{< VIII >}$$

Avec la table de la distribution **t** de student au niveau $\alpha = 0.05$, on a:

$$t_{\frac{\alpha}{2}}(n-1) = t_{0,025}(46) = 2.01 \qquad \text{< IX >}$$

En comparant < VIII > et < IX >, on voit que la valeur de $\frac{|\overline{d_n}|}{S/\sqrt{n}}$ n'est pas tombée dans la région critique (voir aussi la figure 7-1). On ne rejette pas l'hypothèse H_0. Ceci veut dire qu'il n'y a pas de différence signficative entre les deux méthodes de comptage.

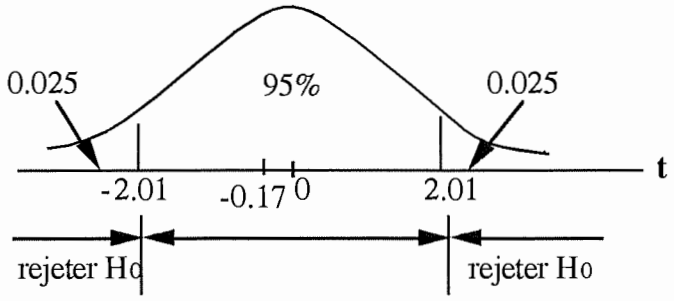

Figure 7-1 Une courbe de la distribution t(student)

Nous sommes conscients que le test précédent ne porte que sur une **moyenne** des différences. Cependant la valeur de **S** n'est pas grande et il ne semble pas y avoir d'écarts systématiques dans un sens ou dans l'autre, dès lors le test nous semble acceptable.

On pourrait encore affiner le résultat en calculant la probabilité de dépassement ou probabilité d'erreur liée à l'échantillon observé [Noir 82].

La probabilité de dépassement vaut, dans ce cas-ci, la probabilité que la valeur absolue de la moyenne d'échantillon $|\overline{d_{47}}|$ excède (dépasse) 0.17, la valeur absolue de la moyenne expérimentale, sous l'hypothèse nulle $\mu = 0$ (probabilité de dépassement ou niveau d'erreur); on calcule donc:

$$P_r(|\overline{d_{47}}| >= 0.17 \mid \mu = 0)$$

Sous l'hypothèse que **d** est une distribution normale $N(\mu;\sigma^2)$ de moyenne μ et de variance σ^2 inconnues, $\frac{|\overline{d_{47}}|-\mu}{S/\sqrt{47}}$ a une distribution **t** de student à 47 -1 = 46 degrés de liberté; on obtient dès lors :

$$P_r(\frac{|\overline{d_{47}}|}{S/\sqrt{47}} >= \frac{0,17}{1.83/\sqrt{47}})$$

où

$$S^2 = \frac{\sum (d_i - \overline{d_{47}})^2}{47 - 1}$$

est la variance d'échantillon et est égale à 3.36 , Cette probabilité est égale à $2(1 - F_{t46}(0.63)) = 0.52$

Cette probabilité excède nettement 0.05 (valeur maximale admise en général) et on ne peut donc en aucun cas rejeter l'hypothèse nulle $\mu = 0$.

Notons que non seulement nous n'avons aucune raison pour rejeter l'hypothèse nulle, mais que les résultats observés sont même très favorables.

N.B : On pourrait traiter ce problème par des tests non paramétriques, Par exemple, nous avons fait le test de WILCOXON et le résultat du test est aussi favorable.

7.2 Complexité de l'algorithme de comptage des cellules

L'algorithme de comptage des cellules (section 5) est constitué de deux parties: la détection des cellules et la détection du contour. Toutes les deux sont des algorithmes itératifs et s'effectuent l'un sur l'autre. C'est-à-dire que la relation de complexité "C_{com}" du comptage des cellules est la suivante:

$$C_{COM} = C_{dc} * C_{dt} \qquad < I >$$

où:

C_{dc} = la complexité de l'algorithme de détection des cellules.

C_{dt} = la complexité de l'algorithme de détection du contour.

Nous allons maintenant calculer séparément la complexité des deux algorithmes.

C_{dc} est fonction du nombre de pixels qui constituent une image, noté N (= 512*512= 262000), et du nombre de cellules dans l'image, noté M. Notons que pour chaque point d'une cellule, on doit vérifier si ce point a été déjà exploité par la détection du contour. Cela veut dire que dans cette étape, pour chaque pixel d'une cellule, le nombre maximal de vérification est (M-1). Donc, la complexité de l'algorithme de détection des cellules est de l'ordre de

$$C_{dc} = O[N * M*(M-1)] \qquad < II >$$

C_{dt} est fonction du nombre de pixels qui constituent la cellule conservée, noté P, et du nombre de points voisins à choisir pour déterminer le point suivant dans le contour, noté V. Lorsque la détection s'effectue dans huit directions (voir la figure 5-2), la valeur maximale est 8. La complexité de l'algorithme de détection du contour est de l'ordre de

$$C_{dt} = O(P * V) \qquad < III >$$

Donc la complexité de l'algorithme de comptage des cellules est de l'ordre de

$$C_{com} = O[N * M*(M-1)] * O(P * V)$$
$$= O[N * M*(M-1) * P * V] \qquad < IV >$$

Dans la formule < IV >, N et V sont constants, 262000 et 8 respectivement. M varie d'une image à l'autre.

Si on remplace P par N/M (le nombre moyen de points par cellule). l'équation < IV > devient

$$C_{com} = O[N^2 * (M-1) * V] \qquad < V >$$

Or, dans une image, le nombre de cellules est très restreint par rapport à la valeur $N^2 = (262k)^2$. A savoir, M est de l'ordre de 20 par image.

On peut le négliger, de même que V(=8) peut être omis, la formule < V > devient donc:

$$C_{com} = O(N^2) = 69 * 10^9 \qquad < VI >$$

Selon < VI >, nous remarquons que l'algorithme est complexe et que son calcul nécessite des ressources importantes. C'est-à-dire, si l'on veut une performance raisonnable, il faut une machine plus performante (voir la section 7-3).

7.3 Perspectives de développements

Nous allons maintenant réfléchir aux possibilités d'extension de notre travail. Les aspects intéressants pourraient être les suivants:

- **Convivialité.** Il pourrait être intéressant de permettre aux utilisateurs de sélectionner les menus et fonctionnalités du système sous un "window" à l'aide de la souris. Si tel était le cas, le module d'interface interactive devrait subir certaines modifications.

- **Amélioration de la performance du traitement d'images.** La digitalisation et le traitement des images se fait actuellement par l'utilisation du logiciel de traitement d'images PCSCOPE. L'expérience a montré qu'il est très utile de l'intégrer dans notre application spécifique. Mais, il est aussi vrai que certaines fonctions de traitement d'images offertes par PCSCOPE sont extrêmement inefficaces. Par exemple, si l'on applique un seul filtrage Médian à une image, le temps d'exécution dure plus de 3 minutes! Tandis que lorsqu'on applique un Médian d'un autre système [Grad 92] à la même image, il ne prend que quelques secondes. Donc, il serait intéressant de programmer soi-même certains filtrages afin de rendre le système plus performant.

- **Transplantation du système sur un ordinateur plus puissant.** Nous avons remarqué tout au long du travail que le fonctionnement du système sur l'ordinateur Olivetti 24M a des contraintes considérables. Premièrement, l'espace de stockage est très limité. Nous savons que pour stocker une image à 512 * 512 pixels, il est nécessaire de disposer d'une mémoire d'au moins 256K. Ce qui signifie qu'un Olivetti dont la capacité de mémoire est 20 MB ne peut stocker que 78 images digitalisées sans compter les autres logiciels sur le disque. Donc, il n'est pas possible de faire le comptage automatique pour une boîte de Pétri, en sachant que le quadrillage d'une boîte de Pétri délimite normalement 10 * 10 = 100 champs de cellules. Ce qui représente 100 images.

Deuxièmement, la vitesse de calcul sur Olivetti n'est pas très grande. Nous avons vu, dans la section 7.2 que la complexité des calculs est très élevée. Pour avoir une bonne performance, il faudrait une machine plus puissante.

- Vers un système intelligent d'aide au comptage des cellules des différents types morphologiques

Pour approfondir notre étude et rendre le système plus pratique pour des biochimistes, il faudrait faire le comptage des cellules de différents types morphologiques; c'est-à-dire que le système pourrait déterminer le nombre de cellules qui ont le même type morphologique au cours de leur culture. Ceci peut nous conduire à chercher des méthodes plus complexes de traitement d'images et d'analyse quantitative.

Il nous paraît possible à plus longue échéance de bâtir un système intelligent qui consiste en deux parties:

1) Offrir des outils d'aide à l'interprétation des données obtenues concernant les images de cellules et permettant d'assurer la gestion de ces données et de guider l'utilisateur dans la conduite de son expérience.

2) Un tel système devrait aussi offrir une gestion de plans d'expérience de traitement d'images, et guider l'utilisateur dans la réalisation de comptages des cellules au moyen de ses connaissances des différents types morphologiques. Pour ce faire, il nous semble qu'il faudra en premier lieu développer différents processus de traitement d'images pour les cellules de différents types (voir Figure 1-1). Le processus de traitement de cellules d'un type donné pourrait être différent de celui que nous avons exposé à la section 4.3.

Chapitre 8 :

Conclusions

Le cadre dans lequel le mémoire doit s'intégrer est le développement d'un système d'analyse quantitative par le traitement d'images. Nous avons voulu fournir au laboratoire de biochimie un outil informatique pour compter les cellules en culture. Actuellement, le système ne traite que les cellules en culture dans des conditions normales. Il n'est pas à même de reconnaître les différents types morphologiques, mais nous croyons qu'en ajoutant quelques procédures spécifiques, la possibilité peut être envisagée.

Au fil de ce travail, nous avons conduit le lecteur au travers de la technique des cultures de cellules réalisées au laboratoire de biochimie cellulaire. Nous nous sommes principalement appliqués dans ce travail à décrire notre système, un système prototype d'aide au comptage de cellules. Cela nous a permis en particulier de préciser le principe du traitement d'images à l'aide du logiciel PCSCOPE, et les algorithmes de détection des cellules. Cela conduit, en outre, à entrevoir les caractéristiques techniques que devrait avoir un système futur. Cela suppose toutefois de nombreux investissements à la fois techniques et théoriques. L'amélioration constante de la qualité des images des cellules obtenues et l'adaptabilité de la méthode à différents échantillons de cellules demanderont la génération de nouveaux outils d'aide à l'interprétation des expériences biologiques et informatiques. On

peut penser qu'il devrait faire largement appel aux nouveaux développements de l'intelligence artificielle.

Ce travail a été un essai, mais a eu comme originalité de lancer des ponts entre la biologie et l'informatique. Il a ainsi permis de montrer l'importance que peut avoir l'informatique pour la mise au point de systèmes d'analyse quantitative cellulaire. Nous espérons qu'il fournira dès lors un apport positif pour de futurs développements dans ce domaine.

9. Bibliographie

[Appe 86] **R. Appel, M. Funk, D. Hochstrasser, A.F. Müller,** "MELANIE. Un système expert d'aide à l'analyse et à l'interprétation d'electrophorèses bidimensionnelles " Press.Univ. Nancy. (1986).

[Colt 90] **P. Coltelli and P. Gualtieri,** " A procedure for the extraction of object features in microscope images, J. Biomed. Comput., 25, 3(1990) 169-176.

[Brun 86] **M. Brunner and W. Ittner** " VIPER: a general-purpose digital image-processing system applie to video microscopy " Computer Methods and Programs in Biomedicine, 26 (1988) , 167-182.

[Deho 68] **R.T. Dehoff and F.N. Rhines**, 3 Quantitative microscopy ". McGraw-Hill Book Co., New York, (1968).

[Deho 90] **P.Dehoux** " Traitement d'images par processeurs parallèles ", Mémoire, Faculté des Sciences Sociales, Politiques et Economiques, ULB. Bruxelles, (1990)

[Dill 83] **K.R.Diller and J.M. Knox**, "Automated computer analysis of cell size changes during cryomicroscope freezing a biased trident convolution mask technique" Cryoletters, (1983) 4, 72.

[DILL 87] **K.R. Diller and S.J. Aggarwal,** " Computer automated cell size an shape analysis in cryomicroscopy". Journal of microscopy, (1987), 46 (2), 209.

[Feyt 91] **E. Feytmans** "Statistique appliquée aux sciences biomédicales" Notes de cours, Département de biologie, F.U.N.D.P, Namur, (1991)

[Gilb 90] **N. Gilbert**, "Statistiques " traduit et adapté par J. Savard 2^{e} édition, Edition Etudes Vivantes 8023, Montréal, Canada (1990).

[Gual 91] **P. Gualtieri, P.Coltelli** "An image-processing system, motion analysis oriented (IPS-100), applied to microscopy " Computer Methods and Programs in Biomedicine, 36, 15-25, (1991)

[Gran 92] **V. Granville** "RAINBOW" un logiciel graphique dédicacé"Département mathématique de Faculté des Sciences, F.U.N.D.P, Namur 28 janvier (1992)

[Heni 89] **V. Henin** "Analyse quantitative de gel d'électrophorèse bidimentionnelle par traitement d'image ", Mémoire, Institut d'Informatique, F.U.N.D.P. Namur,(1989)

[Houb 81] **A. Houben**, Thése: "Modifications enzymatiques au cours du vieillissement des cellules en culture mise en évidence du mecanisme d'alteration de la glucose - 6 - phosphate deshydrogenase" Laboratoire de Biochimie Cellulaire, F.U.N.D.P. Namur,(1981).

[HU 89] **Z. P. Hu, T. Pun and C. Pellegrini,** "An expert system for guiding image segmentation", Computerized medical imaging and graphics. vol.14, . N°.1, pp. 13-24, 1(989).

[Jans 91] **D. Janssens,** mémoire: " Effet de l'aluminium sur le vieillissement cellulaire " Facultes des Sciences. Département de Biologie, F.U.N.D.P. Namur,(1991.)

[Land 84] **U. Landegren** "Measurement of cell numbers by means of the endogenous enzyme hexosaminidase. Applications to detection of lymphokines and cell surface antigens "Journal of immunological methods, 67(1984) 379-388.

[Lest 78] **J. M. Lester, H. A. Williams, B. A. Weintraub,** " Two graphic searching techniques for boundary finding in white blood cell images". Comput. Bio. Med. 8:293-308; (1978).

[Lowi 89] **G. E. Lowitz, G. Stamon** "Images, principes, traitements, applications" Douzième Ecole Internationale d'informatique AFCET, F.U.N.D.P, Namur, Belgium, (1982).

[Mich 87] **S. S. Michael, D. J. Dunn** "Direct measurements of cell number using computer aided video microscopy" Analytical Biochemistry 167,(1987)

[Noir 82] **M. Noirhomme-Fraiture** "Introduction à la statistique mathématique" Notes de coursInstitut d'Informatique, F.U.N.D.P, Namur, (1982)

[OPTI 88] **OPTILAB**: "User's manual, Image processing and analysis software for the Apple Macintosh II", Graftek France, (1988).

[Pavl 82] **T. Pavlidis**, "Algorithmes for Graphics and image processing " Computer science press, Rockville MD, (1982).

[PCSC 87] **PC-SCOPE** Manuel d'utilisation, I2S 239, rue du Jardin Public - B.P. 76, F33041 Bordeax, (1987)

[Prat 78] **W.K. Pratt** "Digital image processing" John Wiley&Sons (1978)

[Pres 81] **Jr. Preston**, "Image processing software: a survey". In: Kanal, L.N.; Rosenfeld, A., eds. progress in pattern Recognition. Amsterdam, Holland/ Elsevier Science Publishers B. V.; North Holland; (1981): 123-148.

[Prew 66] **J. M. S. Prewitt, M. L. Mendelsohn**, "The analysis of cell images" Ann. N. Y. Acad. Sci., vol. 128, pp. 1035-1053, (1966)

[Ramm 90] **P. Ramm** "Image analyzers for biomedical applications" Computerized medical imaging and graphics. vol. 14. N°.5, (1990).

[Rema 79] **J. Remacle** "Le vieillissement cellulaire" Faculté des Sciences. Département de Biologie F.U.N.D.P. Namur,(1976).

[Roy 90] **J. N. Roy, P. Steiger and C. Clifton Ling** "Overlap detection and contour-tracking algorithms for critical organs-application to kidney" Computerized medical imaging and graphics. vol.14, N°.3, pp. 153-161, (1990).

[SCHÖ 89] **M. Schönfeld and R. Grebe** "Automatic shape quantification of freely suspended red blood cells by isodensity contour tracing and tangent counting " Computer Methods and Programs in Biomedicine, 28(1989) , 217-224.

[Seit 83] **P. Seitz, P. Ruegsegger;** " Fast contour detection algorithm for high precision quantitative" IEEE Transactions Med. imag. Vol. M1-2, N°. 3; (1983).

[Serr 82] **J. Serra,** "Image analysis and mathe matical morphology" Academic press; (1982).

[SPID 83] **SPIDER** User's manual, JSD joint system development corp. (1983).

[Tous 91] **O. Toussaint** "Rapport d'activité" Laboratoire de Biochimie Cellulaire, F.U.N.D.P. Namur, (1981).

[Vinc 87] **P. Vincens** "Analyse informatisée des images de gels d'électrophorèse bidimensionnelle" E.N.S Laboratoire de Zoologie, 46, rue d'Ulm Paris, (1987).

[Wang 90] **J. Z. Wang, R. S. Mezrich and D. A. Sebok** "Image processing on macintosh II : A practical boundary finding algorithm for biomedical measurement" Computerized medical imaging and graphics. vol.14, N°.3, pp. 163-171, (1990).

[Weib 79] **E.R. Weibel,** "Stereological methods. Vol. 1. Practical methods for biological morphometry". Academic press, London, (1979).

10. Annexes

Annexe B:

Vocabulaire

Image originale: une image originale est une image digitalisée n'ayant pas encore été soumise à un traitement spécifique donné.

Pixel: le pixel ("picture élément") est le plus petit élément d'une image digitalisée.

convolution: la convolution consiste en un produit d'une valeur discrète par une combinaison de ses éléments voisins.

Binarisation: la binarisation est le résultat d'une transformation d'image digitalisée, d'un système quelconque vers un système de base deux.

Gradient: le gradient est un outil mathématique qui permet de faire ressortir la variation entre différentes valeurs d'éléments.

Annexe B

Modélisation de la recherche du seuil de binarisation

Supposons que les pixels des objets auxquels nous nous intéressons se distribuent normalement dans une image, et que les niveaux de gris de l'image varient dans l'intervalle $[Z_1, Z_2, \ldots Z_k]$.

Soit $P_1(Z)$ la probabilité associée à chaque valeur de Z pour les pixels des objets;

Soient μ_1 et σ_1 la moyenne et la variance respectivement.

De même, soit $P_2(Z)$, μ_2 et σ_2 la probabilité associée à chaque valeur de Z pour les pixels du fond, la moyenne et la variance respectivement.

Notre but est de trouver un seuil, S qui sépare les objets et le fond de l'image avec la probabilité des erreurs minimales.

Soit θ le pourcentage de pixels des objets dans l'image; et $1-\theta$ celui de pixels du fond.

Une distribution normale et mélangée à deux dimensions d'une image peut être représentée comme suit:

$$P(Z) = \theta P_1(Z) + (1 - \theta)P_2(Z) =$$

$$= \frac{\theta}{\sqrt{2\pi}\sigma_1} e^{\frac{-(z-\mu_1)^2}{2\sigma_1^2}} + \frac{\theta}{\sqrt{2\pi}\sigma_2} e^{\frac{-(z-\mu_2)^2}{2\sigma_2^2}}$$

Une fois que l'on a choisi S, la probabilité de mal choisir et de prendre les pixels d'objets comme pixels du fond est:

$$E_1(S) = \int_S^{\infty} P_1(Z)dz.$$

De même, la probabilité de mal choisir et de prendre les pixels du fond comme pixels des objets est:

$$E_2(S) = \int_S^{\infty} P_2(Z)dz.$$

Or, la probabilité totale d'erreurs est

$$E(S) = \theta E_1(S) + (1 - \theta)E_2(S)$$

Alors, $$\ln\frac{\theta\sigma_1}{(1-\theta)\sigma_1} - \frac{(S-\mu_1)^2}{2\sigma_1^2} = \frac{(S-\mu_2)^2}{2\sigma_2^2}$$

Quand $\sigma_1^2 = \sigma_2^2 = \sigma^2$, on a

$$S = \frac{(\mu_1+\mu_2)}{2} + \frac{\sigma^2}{(\mu_2-\mu_1)} \ln\frac{\theta}{(1-\theta)}$$

Soit $\theta = 1/2$

$$S = \frac{(\mu_1+\mu_2)}{2}$$

N.B: Le modèle décrit ci-dessus est seulement applicable aux images dont l'histogramme présente les deux pics.

Annexe C:

Spécification des procédures

PROCEDURE : **acboit ( )**

MODULE : Acquisition des données

PARAMETRES D'ENTREE :

PARAMETRES SORTIE : nomop = pointeur sur la chaîne de caractères contenant le nom de l'opérateur
type: pointeur

DESCRIPTION : Cette procédure réalise l'acquisition des images.

APPEL PRELIMINAIRE : acq-image( )

PROCEDURE : **Acq-image( )**

MODULE : Acquisition des données

PARAMETRES D'ENTREE :

PARAMETRES SORTIE : fich.fim : le fichier contenant l'image digitalisée correspondant à un champ de cellules d'une boîte de Pétri
type: pointeur

symb.p : le fichier contenant les informations symboliques qui sont nécessaires pour le traitement des images suivantes
type: pointeur

DESCRIPTION : Cette procédure réalise l'acquisition automatique de façon interactive. Elle est utilisée pour saisir les images correspondant aux champs de cellules tracés dans une boîte de Pétri.

APPEL PRELIMINAIRE :

PROCEDURE : **balayage(** x0, y0, dx, dy, fichnom)

MODULE : Comptage des données

PARAMETRE D'ENTREE : x0, y0 = coordonnées du point supérieur gauche de l'image traitée.
type: entier

dx = nombre de colonnes de l'image à traiter.
type: entier

dy = nombre de lignes de l'image à traiter.
type: entier

fichnom = pointeur sur la chaîne de caractères contenant le nom du fichier de l'image traitée.
type: pointeur

PARAMETRE DE SORTIE : x0, y0 = position en x et y du point d'une cellule. Ce point est le point de départ du contour de la cellule à détecter.

DESCRIPTION : Cette procédure détecte les cellules possibles dans une image donnée Une fois que le premier point de la cellule est détecté, la procédure appelle la procédure détec-contour(), pour tracer le contour de la cellule.

APPEL PRELIMINAIRE : L'image traitée

PROCEDURE : **Binfir**(x0, y0, dx, dy, xr, yr, seuilp)

MODULE: Traitement d'images

PARAMETRES D'ENTREE: x0, y0 = coordonnées du point supérieur gauche de l'image à traiter
type: entier

xr, yr = coordonnées du point supérieur gauche de l'image résultat (après avoir passé l'opération binarisation).
type: entier

dx = nombre de colonnes de l'image à traiter.
type: entier

dy = nombre de lignes de l'image à traiter.
type: entier

seuilp = seuil de binarisation
type: caractère(niveau de gris)

PARAMETRES SORTIE :

DESCRIPTION : binarisation de l'image afin de séparer une image en deux régions différentes, l'objet en blanc et le fond en noir.

APPEL PRÉLIMINAIRE : Binarisation de l'image

PROCEDURE : **Charg_fich_sym**(comment, colchamp, ligchamp, typecell)

MODULE : Accès-extra BD

.

PARAMETRES D'ENTREE : nomop = le pointeur sur la chaîne de caractères contenant le nom de l'opérateur.
type: pointeur

nomficel = le pointeur sur la chaîne de caractères contenant le nom du fichier à ouvrir.
Type : pointeur

Date = pointeur sur la structure contenant la date du système (MS-DOS) au moment de l'opération.
type: pointeur

comment = pointeur sur la chaîne de caractères recevant le commentaire à inclure dans le fichier *symb.p* (60 caractères maximum).
Type: pointeur

colchamp = pointeur sur la position de la colonne où le champ se situe.
Type = pointeur

ligchamp = pointeur sur la position de la ligne où le champ se situe.
Type = pointeur

typecell = entier pour indiquer le type d'image à traiter.
0 : image à densité de cellules faible.
1 : image à densité de cellules importante.

PARAMETRES SORTIE : sympout = structure des informations symboliques correspondant à un enregistrement du fichier*symb.p*

DESCRIPTION : Cette procédure réalise le chargement des images digitalisées dans les fichiers de type "*.fim*" et des informations relatives aux images acquises dans le fichier*symb.P*

N.B: Les paramètres qui ne sont pas définis dans la procédure sont les paramètres globaux.

PROCEDURE : **comtra( )**

MODULE : Comptage des cellules

PARAMETRE D'ENTRÉE : fichnom = le fichier de type.fim correspondant à l'image à traiter.
type: pointeur

PARAMETRE DE SORTIE : cellu = le nombre de cellules dans l'image.
type: pointeur

maxmin.p = le fichier qui contient les valeurs maximales ou minimales des cellules détectées dans l'image traitée.
type: pointeur

cell.p = le fichier qui contient l'ensemble des pixels des contours cellulaires d'une image.
type: pointeur

DESCRIPTION : Cette procédure regroupe les modules de traitement d'images et de comptage de cellules afin de réaliser l'automatisation du traitement des images originales acquises par la procédure d'acquisition d'images.

APPEL PRÉLIMINAIRE : l'image traitée

PROCEDURE : **contour-trac**(cellfich)

MODULE : Accès-extra BD

PARAMETRE D'ENTREE : cellfich = le fichier (cell.p) regroupant toutes les listes de pixels de cellules dans une image donnée.
type: pointeur

PARAMETRE DE SORTIE : (sur le vidéo) les contours des cellules pour une image donnée.

DESCRIPTION : Cette procédure retrace les contours des cellules qui ont été détectées. Cette procédure est facultative.

APPEL PRELIMINAIRE : l'image traitée

PROCEDURE : **dans-contour(** x0, y0)

MODULE : Comptage des cellules

PARAMETRE D'ENTREE: x0, y0 = la position du pixel dans l'image.
type: entier

VALEUR DE SORTIE : 0 = Point à la position (x0, y0) n'appartenant pas au contour de la cellule.

1 = le point à la position (x0, y0) appartenant au contour de la cellule.

DESCRIPTION : Cette procédure vérifie si le point (x0, y0) appartient au contour cellulaire.

APPEL PRELIMINAIRE : l'image traitée

PROCEDURE : **Detec-contour**(x0, y0)

MODULE : Comptage des cellules

PARAMETRE D'ENTREE : x0, y0 = coordonnées en x et y du point de départ (origine) de la cellule.

PARAMETRE DE SORTIE: une structure de données qui contient les valeurs maximales et minimales x, y de la cellule concernée.

une liste de la structure de données *pixel* qui contient tous les points du contour de la cellule qui vient d'être détectée.
type: pointeur

compp = le compteur des pixels du contour cellulaire qui contient le nombre de pixels du contour de la cellule concernée.

DESCRIPTION : Cette procédure détecte le nombre de pixels du contour d'une cellule à partir d'un point de départ (X0, Y0) sur la cellule. Ce point est donné par la procédure de balayage.

APPEL PRELIMINAIRE :

PROCEDURE : **Dilate2**(x0, y0, dx, dy, xr, yr, couleur)

MODULE: Traitement d'images

PARAMETRES D'ENTREE: x0, y0 = coordonnées du point supérieur gauche de l'image à traiter
type: entier

xr, yr = coordonnées du point supérieur gauche de l'image résultat (après l'opération de dilatation).
type: entier

dx, dy = dimensions de l'image à traiter.
type: entier

couleur = " couleur " de l'objet

0 : noir (niveau de gris = 0)

1 : blanc (niveau de gris = 255)
type: entier

PARAMETRES SORTIE :

DESCRIPTION : dilatation morphologique sur une image binarisée: procède au remplissage de la cellule.

APPEL PRELIMINAIRE : Binarisation de l'image

PROCEDURE : **Encadre** (x, y, dx, dy)

MODULE : Acquisition des données

PARAMETRES D'ENTREE : &x, &y = pointeur sur les entiers qui indiquent les coordonnées du point superieur gauche de l'image à saisir.
Type: pointeur

&dx = pointeur sur les entiers qui indiquent le nombre de colonnes de l'image à saisir.
Type: pointeur

&dy = pointeur sur les entiers qui indiquent le nombre de lignes de l'image à saisir.
Type: pointeur

PARAMETRES SORTIE : x, y = la position du point original de l'image acquise.
type: entier

dx = le nombre de colonnes de l'image acquise.
type: entier

dy = le nombre de lignes de l'image acquise.
type: entier

DESCRIPTION : Cette procédure permet à l'utilisateur de choisir sur l'écran vidéo et grâce à la souris, la dimension du champ à analyser. Après la sélection du champ par la souris, (on clique) la dimension du champ est mise en mémoire

APPEL PRELIMINAIRE : pris-image( )

PROCEDURE : **Erode**(x0, y0, dx, dy, xr, yr,couleur)

MODULE: Traitement d'images

PARAMETRES D'ENTREE: x0, y0 = coordonnées du point supérieur gauche de l'image à traiter.
type: entier

xr, yr = coordonnées du point supérieur gauche de l'image résultat (après l'opération d'Erosion).
type: entier

dx, dy = dimensions de l'image à traiter
type: entier

couleur = " couleur " de l'objet
0 : noir (niveau de gris = 0)
1 : blanc (niveau de gris = 255)
type: entier

PARAMETRES SORTIE :

DESCRIPTION : Erosion morphologique sur une image binarisée.

APPEL PRELIMINAIRE : Binarisation de l'image de cellules

PROCEDURE : **fcell(** tetelist, cellu)

MODULE : Accès-extra BD

PARAMETRES D'ENTREE : tetelist = pointeur sur la tête de la liste des pixels du contour cellulaire.
type: pointeur

cellu = le nombre de cellules dans l'image concernée.
type: pointeur

PARAMETRES SORTIE : cell.p = le fichier qui contient toutes les listes des pixels des contours cellulaires dans un champ de l'image
type: pointeur

DESCRIPTION : Cette procédure réalise l'enregistrement des listes de pixels sur les contours cellulaires dans le fichier *cell.p*.

APPEL PRELIMINAIRE :

PROCEDURE : **Fmaxmin**(xm, ym, xa, ya)

MODULE: Accès-extra BD

PARAMETRES D'ENTREE: xm = la valeur minimum de la colonne x d'une cellule concernée
type: entier

ym = la valeur minimum de la ligne y d'une cellule concernée.
type: entier

xa = la valeur maximum de la colonne X d'une cellule concernée.
type: entier

ya = la valeur maximum de la ligne Y d'une cellule concernée.
type: entier

PARAMETRES SORTIE : minmax = un tableau de l'ordre de 4 qui contient xmin, ymin, xmax, ymax du contour de la cellule concernée.

DESCRIPTION : Cette procédure met les quatre valeurs: xm, ym, xa, ya dans un tableau qui est stocké dans le fichier *maxmin.p* .

APPEL PRELIMINAIRE : l'image traitée

PROCEDURE : **Initfgb( )**

MODULE : Traitement d'images

PARAMETRES D'ENTREE :

PARAMETRES SORTIE :

DESCRIPTION : Cette procédure initialise la carte mémoire image, la mémoire vidéo, prépare le hardware pour l'acquisition d'image

APPEL PRELIMINAIRE : Etat initial de la carte:

- groupe de LUTs actives = 0
-caméra = 1
-standard TV européen
-synchronisation externe
-quadrant visualisé = 0
-mode 4 quadrants
-toutes les LUTS sont initialisées à rampe croissante
- offset = 120
- gain = 80

PROCEDURE : **Maxmin**(xmi, ymi, xma, yma, xk, yk)

MODULE : comptage des cellules

PARAMETRE D'ENTREE : xmi = la valeur minimale de la colonne x du contour cellulaire x0 étant le point de départ pour la détection du contour.
type: entier

ymi = la valeur minimale de la ligne y du contour cellulaire.
Originalement, elle est égale à Y0.
type: entier

xma = la valeur maximale de x au niveau du contour cellulaire.
type: entier

yma = la valeur maximale de y au niveau du contour cellulaire.
type: entier

xk, yk = le point courant de la détection du contour.
type: entier

PARAMETRES DE SORTIE :

DESCRIPTION : Cette procédure permet de comparer le point courant (xk, yk), aux points maximaux (xma,yma) et minimaux (xmi, ymi).
si xk < xmi, xmin est remplacé par xk,
si xk > xma, xma est remplacé par xk.
Même chose qu'avec xk, xmi et xma, et yk, pour ymin et yma.

APPEL PRELIMINAIRE :

PROCEDURE : **Median**(x0, y0, dx, dy, xr, yr)

MODULE: traitement d'images

PARAMETRES D'ENTREE: x0, y0 = colonne et ligne de départ de l'image
type: entier

xr, yr = coordonnées du point supérieur gauche de l'image résultat (après filtrage médian).
type: entier

dx = nombre de colonnes de l'image à traiter.
type: entier

dy = nombre de lignes de l'image à traiter.
type: entier

PARAMETRES SORTIE :

DESCRIPTION : filtrage médian 3x3 d'une image, pour éliminer les bruits de fond.

APPEL PRELIMINAIRE :

PROCEDURE : **Menu_prin()**

MODULE: Interface interactive

PARAMETRES D'ENTREE:

PARAMETRES SORTIE :

DESCRIPTION : Cette procédure affiche le menu principal. Quand l'utilisateur frappe sur le clavier un numéro, la procédure appelle la procédure correspondante pour réaliser la demande de l'utilisateur.

APPEL PRELIMINAIRE :

PROCEDURE : **Point-exp**(xs, ys)

MODULE : comptage des cellules

PARAMETRE D'ENTREE: xs = position en x d'un pixel de l'image cellulaire traitée.
type: entier

ys = position en y d'un pixel de l'image cellulaire traitée.
type: entier

VALEUR DE SORTIE : 0 = le point (xs,ys, gris = 255) n'a pas encore été exploité.

1 = ce point a déjà été exploité.

DESCRIPTION : Cette procédure vérifie si le point (xs, ys) de la cellule appartient déjà au fichier *maxmin.p*. Si oui, la procédure retourne une valeur 1, Sinon une valeur 0.

APPEL PRELIMINAIRE : l'image traitée

PROCEDURE : **Pris-image**(nomficel, comment)

MODULE : Acquisition des données

PARAMETRES D'ENTREE : fichnom = pointeur sur la chaîne de caractères contenant le nom d'un champ de cellules.
Type: pointeur

comment = pointeur sur la chaîne de caractères contenant la dimension de champ
Type: pointeur

PARAMETRES SORTIE : un fichier digitalisé de type ".fim " au nom "*fichnom*".

DESCRIPTION : Cette procédure permet de digitaliser l'image d'un champ de cellules cultivées dans une boîte de Pétri. Le nom de l'image digitalisée est contenu dans *fichnom*.

APPEL PRELIMINAIRE : savechamp ( )

PROCEDURE : **Sauvechamp**(idenboit)

MODULE : Acquisition des données

PARAMETRES D'ENTREE: idenboit = pointeur sur la chaîne de caractères contenant l'identificateur de la boîte de Pétri.
Type: pointeur

PARAMETRES SORTIE : nomficel = pointeur sur la chaîne de caractères contenant le nom de fichier symbolique.
Type: pointeur

comment = pointeur sur la chaîne de caractères recevant le commentaire à inclure dans le fichier symbolique (60 caractères maximum)
Type: pointeur

nomcham = pointeur sur la chaîne de caractères contenant le nom du l'opérateur
Type: pointeur

colchamp = pointeur sur la position de colonne où le champ se situe
Type: pointeur

ligchamp = pointeur sur la position de la ligne où le champ se situe
Type: pointeur

typecell = entier pour indiquer le type de l'image à traiter

0: image à densité de cellules faible.
1: image à densité de cellules importante.

DESCRIPTION : Cette procédure réalise l'acquisition des informations symboliques. Elle utilise les procédures pris-image() et char-fichsym() pour réaliser la digitalisation de l'image et l'enregistrement des informations symboliques.

PROCEDURE : **Sobfir**(x0, y0, dx, dy, xr, yr)

MODULE: **traitement d'images**

PARAMETRES D'ENTREE: x0,y0 = coordonnées du point supérieur gauche de l'image à traiter.
type: entier

xr, yr = coordonnées du point supérieur gauche de l'image résultat (après filtrage sobel).
type: entier

dx, dy = dimensions de l'image contenant les cellules à compter.
type : entier

PARAMETRES SORTIE :

DESCRIPTION : Cette procédure permet d'augmenter le contraste au niveau du contour des objets Gradient du sobel en XY.
Gx et Gy sont calculées. Alors le gradient du point(x, y) est donné par:
| Gx | + | Gy |.

PROCEDURE : **Trouve-seuil**(x, y, dx, dy, seuilp)

MODULE : traitement d'images

PARAMETRES D'ENTREE : x, y = les entiers qui indiquent les coordonnées du point supérieur gauche de l'image à traiter.
type: entier

dx, dy = les entiers qui indiquent la dimension de l'image à traiter.
type: entier

seuilp = pointeur sur l'entier qui contient le seuil de binarisation pour l'image à traiter
type: pointeur

PARAMETRES SORTIE : seuilp = l'entier contenant le seuil choisi.
type: entier

DESCRIPTION : Cette procédure a pour but de calculer l'histogramme de l'image à traiter et de choisir un seuil de binarisation

APPEL PRELIMINAIRE :

Annexe D

Les sources des programmes

```
 #include <stdio.h>
 #include <stdlib.h>
 #include <memory.h>
 #include <math.h>
 #include <string.h>
 #include <ctype.h>
 #include <dos.h>
 #include <graph.h>

struct symboly {
         char   nomfisym[9];
         char   champcol[3];
         char   champlig[3];
         struct dosdate_t  date;
         char   nomoper[20];
         int    typecell;
         int    nombcell;
};

struct list_pixel {
          int colonne;
          int ligne;
          struct list_pixel *next;
}

typedef struct list_pixel pixcont;
typedef pixcont  *link;


 /***************************** acqautcm ************************/

#include "prog.h"

extern int symsize;
extern char nomop[20];

typedef struct symboly symcont;
typedef symcont *sympoint;

/* ********************************************************************

*  Sauvegarder l'image dans le fichier disque de type *.fim       *

******************************************************************** */

void pris_image(nomficel, comment)
char *nomficel, *comment;
/* nomficel : contenant le nom du fichier  */
/* x,y : le point de départ de l'image     */
/* dx, dy : le dimension de l'image        */
{
 int x, y, dx, dy;

 initfgb();
 presetfgb(0);
 videoeur();
 selnumcam(3);
 syncext();
 initfgb();
 syncext();
 cacquire();
 encadre(&x, &y, &dx, &dy);
 printf("\ntaper un clavier pour saisir l'image\n");
 inkeyb();
 frmfreez();
 delay(1);
 diskwrit(nomficel, comment, x, y, dx, dy, 'C');
 curseur(52, 28);
```

```
 ecrire(nomficel);
}

/* ************************************************************
*  Les procedures dans le module d'acquisition de donnees     *
************************************************************ */

void encadre(x, y, dx, dy)
int *x, *y, *dx, *dy;

{
  int bstat, xpos, ypos, xdeb, ydeb,  xfin, yfin, cadre;
  char libelle[2];

  cadre = 1;
  xdeb = 100;
  ydeb = 100;
  M_GETPOS(&bstat, &xpos, &ypos, xdeb, ydeb, 255, 10, -1, libelle);
  delay(1);
  xdeb = xpos;
  ydeb = ypos;
  while (cadre == 1) {
  M_GETCADRE(&bstat, &xfin, &yfin, &xdeb, &ydeb, 255, 10, -1, libelle
  delay(1);
  *x = xpos; *y = ypos;
  *dx = xfin - xdeb;
  *dy = yfin - ydeb;
  if ((*dx > 0) && (*dx < 511) && (*dy > 0) && (*dy < 511))
  break;
}
  printf("\nx=%d, y=%d, dx=%d, dy=%d", *x, *y, *dx, *dy);
}

/* ************************************************************
*   Charger les informations symboliques dans le fichier "symb.p"   *
************************************************************ */

void charg_fich_sym(nomficel, comment, colchamp, ligchamp, typecell)
char *nomficel, *comment, *colchamp, *ligchamp;
int *typecell;

{
 int i;
 struct dosdate_t date;
 sympoint   smpoint;
 FILE *fp1, *fopen();

 _dos_getdate(&date);
 smpoint = (sympoint) malloc(sizeof(symcont));
 strcpy(smpoint -> nomfisym, nomficel);
 strcpy(smpoint -> champcol, colchamp);
 strcpy(smpoint -> champlig, ligchamp);
 smpoint -> date.month = date.month;
 smpoint -> date.day = date.day;
 smpoint -> date.year = date.year;
 strcpy(smpoint -> nomoper, nomop);
 smpoint -> typecell = *typecell;
 smpoint -> nombcell = 0;
 fp1 = fopen("symb.p", "ab");
 if (fwrite(smpoint, sizeof(symcont), 1, fp1))
    symsize++;
 else
 {
```

```
      printf("writing file error\n");
      exit(0);
    }
     free(smpoint);
     fclose(fp1);
  }

  /* ****************************************************************

  *          saisir les informations symboliques                    *
            concernant les champs de la boite de Petri

  **************************************************************** */

  void savechamp(idenboit)
  char *idenboit;

  {
   int typecel;
   char c, nomficel[9], comment[40], nomchamp[3], colchamp[3], ligchamp

   strcpy(nomficel, idenboit);
   printf("\ndonner un identificateur de l'image S.V.P; (p.ex. 01:) \n"
   scanf("%s", nomchamp);
   printf("et le type de cellule (0,1,2,3) ");
   scanf("%d", &typecel);
   strcat(nomficel, nomchamp);
   strcat(nomficel, ".fim");
   strcpy(comment, "L'identificateur de la boîte = ");
   strcat(comment, idenboit);
   printf("\ndonner la location du champ, S.V.P. (x y) ");
   scanf("%s%s", colchamp, ligchamp);
   charg_fich_sym(nomficel, comment, colchamp, ligchamp, &typecel);
   printf("ajuster le microscope et choisir l'encadrement ");
   printf("à l'aide de la souris: \n");
   printf("si ok, taper une bouton de la souris;\n");
   pris_image(nomficel, comment);
   printf("\nvous voulez sauver l'autre champ de cellules? (y/n)  ");

  }


  /* ****************************************************************

  *    saisir les informations concernant de la boite de Petri       *
     _ une boucle pour saisir les images correspond aux
      plusieurs champs de cellules dans cette boite

  **************************************************************** */

  void ac_boit()
{
    char respchamp, respboit, nomboit[4];

    printf("\nvous voulez sauver la boite de Petri (y/n)?  ");
    for (;;) {
      respboit = getchar();
      scanf("%c", &respboit);
      if (respboit == 'y')
   {
      printf("\nentrer un identificateur de la boite, p.ex. YI:  ");
      scanf("%s", nomboit);
      printf("\nallez vous sauver le champ de celluels (y/n)");
   for(;;)
   {
      respchamp = getchar();
      /* scanf("%c", &respchamp);  */
      if (respchamp == 'n')
           break;
      else if (respchamp == 'y')
```

```
            savechamp(nomboit);
  }
  break;
  }
 }
}


/* ****************************************************************

*        La procedure principale de l'acquisition de doneees        *

**************************************************************** */

acboit_e()
{
/* symsize = 0; */
_clearscreen(_GCLEARSCREEN);
printf("*************************************************************");
printf("\n* Bienvenue au programme d'acquisition d'images !     *");
printf("\n*************************************************\n")
ac_boit();
printf("fin de l'acquisition");
}

/* compticm.c */
#include "prog.h"
#include "pixlist.h"

/* #define MIN(a,b) (a < b) ? a : b
#define MAX(c,d) (c > d) ? c : d    */

/* ****************************************************************

* Le comptage de cellules
*     - le choix du seuil de binarisation
*     - le traitement d'images
*     - la detection des cellules
*     - la detection du contour des cellules
*     - retracage des contours des cellules de l'image

**************************************************************  */


typedef struct { int col;
         int lign;
         int numcell;
         } pixel;

int filesize;
int cellu, cellnoy;
link listete, listqueu;


/********* procedures dans le module d'access-extra donnees *********

void mod_image(x0, y0, dx, dy)
int x0, y0, dx, dy;

{
  int x, y;
  unsigned char gris0;

  gris0 = 0;
  for(y=y0; y < y0 + 10; y++)
    for (x = x0; x <= x0 + dx; x++)
         putpixel(&gris0, x, y);
    for (x = x0; x < x0 + 10; x++)
        for(y = y0 + 10; y <= y0 + dy; y++)
           putpixel(&gris0, x, y);
```

```
            for(y = y0 + dy - 10; y < y0 + dy; y++)
            for (x = x0 + 10; x <= x0 + dx; x++)
      putpixel(&gris0, x, y);
      for (x = x0 + dx - 10; x <= x0 + dx; x++)
        for(y = y0 + 10; y <= y0 + dy - 10; y++)
            putpixel(&gris0, x, y);
    }


    /*************      Etablir une liste de pixels      ***************
                       sur le contour de la cellule detectee

    void ajoulist(queulist, x, y)
    link queulist;
    int  x, y;

    {
      link  q1;

      q1 = (link) malloc(sizeof(pixcont));
      if (queulist == NULL)   /* premier pixel du contour */
    {
      listete = q1;
      q1 -> colonne = x;
      q1 -> ligne = y;
      q1 -> next = NULL;
    }
else
{
     q1 -> colonne = x;
     q1 -> ligne = y;
     q1 -> next = NULL;
     queulist -> next = q1;
    }
    listqueu = q1;
    }

    /*********** stocker la liste de pixels dans le fichier cell.p ******

    void fcell(tetelist, cellu1)
    link tetelist;
    int  cellu1;

    {
      link  q1;
      pixel pixptk;
      FILE *fp1, *fopen();
      q1 = tetelist;
      fp1 = fopen("cell.p","ab");
      while (q1 != NULL )
      {
        pixptk.col = q1 -> colonne;
        pixptk.lign = q1 -> ligne;
        pixptk.numcell = cellu1;
        if ( fwrite (&pixptk, sizeof(pixptk), 1, fp1))
        filesize++;
       else
        {
            printf("writing file error\n");
            exit(0);
        }
        q1 = q1 -> next;
       }
       fclose(fp1);

    }


    /*****************************************************************
     *                    stocker les points
```

```
                    maxmaux(xm,ym) et maximaux(xa ,ya)                *
 ***************************************************************/

void fmaxmin(xm, ym, xa, ya)
int xm, ym, xa, ya;

{
  int minmax[4];
  FILE *fp2, *fopen();

  fp2 = fopen("maxmin.p", "ab");
  minmax[0] = xm;
  minmax[1] = ym;
  minmax[2] = xa;
  minmax[3] = ya;
  fwrite(minmax, sizeof(minmax), 1, fp2);
  fclose(fp2);

}


/* ************************************************************

*  les procedures dans le module du comptage des cellules    *
   dans_contour: verifier si le point de la détection est
   sur le contour de la cellule

************************************************************/

int dans_contour(xs, ys, xkc, ykc, k)
int *xs, *ys;
int k, xkc, ykc;

{
  unsigned char griss;

  switch( k )
{
   case 0: {
             *xs = xkc + 1; *ys = ykc;
             getpixel(&griss, *xs, *ys);
           }
             break;

   case 1: {
             *xs = xkc + 1; *ys = ykc - 1;
             getpixel(&griss, *xs, *ys);
           }
             break;

   case 2: {
             *xs = xkc; *ys = ykc - 1;
             getpixel(&griss, *xs, *ys);
           }
             break;

   case 3: {
             *xs = xkc - 1; *ys = ykc - 1;
             getpixel(&griss, *xs, *ys);
           }
             break;

   case 4: {
             *xs = xkc - 1; *ys = ykc;
             getpixel(&griss, *xs, *ys);
           }
             break;
   case 5: {
             *xs = xkc - 1; *ys = ykc + 1;
             getpixel(&griss, *xs, *ys);
```

```
            }
              break;
   case 6: {
              *xs = xkc; *ys = ykc + 1;
              getpixel(&griss, *xs, *ys);
            }
              break;
   case 7: {
              *xs = xkc + 1; *ys = ykc + 1;
              getpixel(&griss, *xs, *ys);
            }
              break;
}
 if (griss == 255) return(1);
 else return(0);

}


/* ********************************************************

*   le point courant(xk, yk) est compare avec les points
    minimaux [xmi, ymi] et maximaux [xma, yma]            *

******************************************************** */

maxmin(xmi, ymi, xma, yma, xk, yk)
int  *xmi, *ymi, *xma, *yma;
int xk, yk;

{
  if (xk < *xmi) *xmi = xk;
  else
   if(xk > *xma) *xma = xk;

  if(yk < *ymi) *ymi = yk;
  else
    if(yk > *yma) *yma = yk;
 }


/* ************************************************************

*   la procedure principale de la detection du contour       *

********************************************************** */

detec_contour(x0, y0)
int x0,y0;

{
  int xk, yk, xb, yb, xmin, xmax, ymin, ymax, k1, k2, first, found, c
  static int K;
  unsigned char grisk;
  pixel pixptk;

  xmin = x0; xmax = x0 + 1;
  ymin = y0; ymax = y0 + 1;
  cellnoy++;
  xk = x0; yk = y0;
  K = 6;
  first = 1;
  compp = 0;
  listqueu = NULL;
  while (!((xk == x0) && (yk == y0)) || (first == 1))
 {
  found = 0;
  while (found == 0)
 {
  if ((K - 1) < 0) k1 = 8 + (K - 1);
```

```
  else  k1 = K - 1;
  if ((K + 1) > 7) k2 = K + 1 - 8;
  else k2 = K + 1;
  if (dans_contour(&xb, &yb, xk, yk, k1))
  {
    xk = xb; yk = yb;
    K = K - 2;
    if (K < 0) K = K + 8;
    maxmin(&xmin, &ymin, &xmax, &ymax, xk, yk);
    found = 1;
    compp++;
    ajoulist(listqueu, xk, yk);
  }
    else if (dans_contour(&xb, &yb, xk, yk, K))
         {
          xk = xb; yk = yb;
          maxmin(&xmin, &ymin, &xmax, &ymax, xk, yk);
          found = 1;
          compp++;
          ajoulist(listqueu, xk, yk);
         }
         else if (dans_contour(&xb, &yb, xk, yk, k2))
                {
                 xk = xb; yk = yb;
                 maxmin(&xmin, &ymin, &xmax, &ymax, xk, yk);
                 found = 1;
                 compp++;
                 ajoulist(listqueu, xk, yk);
                }
                else {
                     K = K + 2;
                     if (K >= 8) K = K - 8;
                    }
    }
    first = 0;
 }
  fmaxmin(xmin, ymin, xmax, ymax);
  if (compp <= 30)
    free(listete);
  else
    {
        cellu++;
        fcell(listete, cellu);
        free(listete);
    }
 /*  printf("every thing is going well\n"); */
}

/* ********************************************************

*   verifier si le point (x, y) est deja exploite ou pas  *

******************************************************** */

point_exp(x, y)
int x, y;
{
  FILE *fp2, *fopen();
  int i, mima[4];

  if ((fp2 = fopen("maxmin.p", "rb")) == NULL)
{
 printf("\n cannot open this file");
 exit(0);
}
  for(i = 1; i <= cellnoy; i++)
{
  if(!(fread(mima, sizeof(mima), 1, fp2)))
  if (feof(fp2))
     {
```

```
          fclose(fp2);
          exit(0);
      }
   else
      printf("file reading error\n");
if(((x >= mima[0]) && (x <= mima[2])) && ((y >= mima[1]) && (y <= mim
   {
     fclose(fp2);
     return(1);
   }
}
   fclose(fp2);
   return(0);
}

/* ************************************************************

*    la procedure principale de la detection des cellules
       dans un champ concerne                                  *

************************************************************ */


balayage(x0, y0, dx, dy)
int x0, y0, dx, dy;

{
   int X, Y, i, j, firstcon;
   unsigned char gripix,bkgrd;

  Y=y0;
  bkgrd=0;
  firstcon = 0;
  mod_image(x0, y0, dx, dy);
  for(Y=y0; Y<y0+dy; Y++)
  {
    X=x0;
    while (X < x0+dx)
  {
    getpixel(&gripix, X, Y);
    if (bkgrd != gripix)
         if ( firstcon == 0 ) {
          detec_contour(X, Y);
          firstcon++;
         }
         else
          if (!(point_exp(X, Y)))
          /*  printf("this point is not in the file, to ftrack\n"); */
             detec_contour(X, Y);
   X++;
   }
  }
}

/* ************************************************************

*     retracer les contours des cellules detectees          *

************************************************************ */

contour_trac()
{
   FILE *fp, *fopen();
   pixel pixelk;
   int i, x1, y1;
   unsigned char gris1;

  selpan(1);
  presetfgb(0);
  if ((fp = fopen("cell.p", "rb")) == NULL)
```

```
{
 printf("cannot open file");
 exit(0);
}
  for(i = 1; i <= filesize; i++)
  { if(fread(&pixelk, sizeof(pixelk), 1, fp))
    {
      x1 = pixelk.col;
      y1 = pixelk.lign;
      gris1 = 255;
      putpixel(&gris1,x1,y1);
    }
 else
   {
     printf("file reading error\n");
     exit(0);
   }
 }
   fclose(fp);
}


/* ********************************************************

 *   les procedures dans le module de traitement d'image   *

******************************************************** */

void trouve_seuil(x, y, dx, dy, seuilp)
int x, y, dx, dy, *seuilp;

{
 float FVAL[8];
 long HISTO[256], HIST;
 char SMIN,SMAX;
 int i;

 bimcopy(x, y, dx, dy, x + 512, y);
 disppos(40, 500);
 setmmlin(HISTO, 512, 0);
 flblkhst(x, y, dx, dy, HISTO, FVAL);
 disphfir(HISTO, -1);
 disppos(40, 500);
 bimcopy(x + 512, y, dx, dy, x, y);
 printf("moyenne:   %f\n", FVAL[3]);
 *seuilp = FVAL[3];
 printf("moyenne = %d", *seuilp);
}

void trait_image(x0, y0, dx, dy)
int  x0, y0, dx, dy;

{
 int seuil;
 char comment[60], cbim;

 trouve_seuil(x0, y0, dx, dy, &seuil);
 sobfir(x0, y0, dx, dy, x0, y0);
 seuil = seuil + 10;
 binfir(x0, y0, dx, dy, x0, y0, seuil);
 median(x0, y0, dx, dy, x0, y0);
 dilate2(x0, y0, dx, dy, x0, y0, 1);
 erode1(x0, y0, dx, dy, x0, y0, 1);
}

/* ************************************************************

*  Lecture de l'image au nom contenu dans le "fichnom" d'une image
   (Module d'acces_extra)
```

```
 ************************************************************************

acq_image(x, y, dx, dy, fichnom)
int *x, *y, *dx, *dy;
char *fichnom;

{
  char comment[60], cbim;

 initfgb();
 presetfgb(0);
 DSKRDHDR(fichnom, comment, x, y, dx, dy, &cbim);
 diskread(fichnom, *x,  *y);
}

/* ************************************************************************

*  Trouver les informations concernant à l'image digitalisee dans
le fichier symbolique. Cette image est subit ensuite une série
de transformations. (Module d'acces_extra)      *

************************************************************************

acq_param(nomfich)
char nomfich[5];

{
 int  ok;
 char verf;

 ok = 1;
 verf = '0';
 while ( ok == 1 )
 {
   printf("\n  donnez le nom de fichier à compter(4 car)" );
   scanf("%s", nomfich);
   printf("\n verifiez les entrées (n/y) ?");
   for(;;) {
   verf = getchar();
   if (verf == 'y')
   { ok = 1;
     break;
   }
   else if (verf == 'n')
   { ok = 0;
     break;
   }
  }
 }
}

int verif_fich(fichnom)
char *fichnom;

{
  FILE *fp2, *fopen();

 if ((fp2 = fopen(fichnom, "rb")) == NULL)
{
 fclose(fp2);
 return(0);
}
else
{
 fclose(fp2);
 return(1);
}
}
```

```
/* ************************************************************

*  la procedure principale du comptage des cellules d'une image

************************************************************/

compind()
{
 int x0, y0, dx, dy;
 char nomfich[5], fichnom[9];

 _clearscreen(_GCLEARSCREEN);
 printf("\n*********************************************\n");
 printf(" *     Bienvenue au programme de comptage     *\n");
 printf("*********************************************\n");
 filesize = 0;
 cellu = 0;
 cellnoy = 0;
 acq_param(nomfich);
 strcpy(fichnom, nomfich);
 strcat(fichnom, ".fim");
 if (verif_fich(fichnom) == 1)
 {
 printf("\n nom de fichier = %s",fichnom);
 acq_image(&x0, &y0, &dx, &dy, fichnom);
 trait_image(x0, y0, dx, dy);
 balayage(x0, y0, dx, dy);
 printf("\n\n*************************************************
 printf("\n* Il y a %d cellules dans le fichier %s          * ",
 cellu, fichnom);
 printf("\n*************************************************
 printf("\n %d points de bruits sont elimines\n", (cellnoy - cellu));
 contour_trac();
 system("DEL MAXMIN.P");
 system("DEL CELL.P");
 }
  else
   printf("le fichier %s n'existe pas !\n", fichnom);

 }

 /* ************************************************************

 *  Les procedures dans le module d'acquisition de donnees
    Encadre : permette à l'utilisateur de choisir l'encadrement
    de l'image à l'aide de la souris                          *

************************************************************ */

/* acqindcm.c */
#include "prog.h"

int symsize;
char nomop[20];

void encadrei(x, y, dx, dy)
int *x, *y, *dx, *dy;

{
  int bstat, xpos, ypos, xdeb, ydeb,  xfin, yfin;
  char libelle[2];
  xdeb = 100;
  ydeb = 100;
  M_GETPOS(&bstat, &xpos, &ypos, xdeb, ydeb, 255, 10, -1, libelle);
  delay(1);
  xdeb = xpos;
  ydeb = ypos;
  M_GETCADRE(&bstat, &xfin, &yfin, &xdeb, &ydeb, 255, 10, -1, libelle
  delay(1);
  *x = xpos; *y = ypos;
```

```
  *dx = xfin - xdeb;
  *dy = yfin - ydeb;
  printf("x=%d, y=%d, dx=%d, dy=%d\n", *x, *y, *dx, *dy);
}

/* Sauvegarder le champ d'une boite de Petri */
void savechampi(adficel)
char adficel[4];

{
   int x, y, dx, dy;
   char  nomchamp[3], nomficel[9];

   strcpy(nomficel, adficel);
   printf("donner un identificateur du champ de cellules, S.V.P; p.ex
   scanf("%s", nomchamp);
   strcat(nomficel, nomchamp);
   initfgb();
   presetfgb(0);
   videoeur();
   selnumcam(3);
   syncext();
   initfgb();
   printf("\n si ok, taper une bouton de la souris;");
   syncext();
   cacquire();
   encadre(&x, &y, &dx, &dy);
   printf("\n ajuster le microscope, si ok, taper un clavier; S.V.P."
   inkeyb();
   frmfreez();
   strcat(nomficel, ".fim");
   diskwrit(nomficel, " ", x, y, dx, dy, 'C');
   curseur(52, 28);
   ecrire(nomficel);
   printf("\nvous voulez sauver l'autre champ de cellules? (y/n)  \n"
 }


 /* saisir les images correspond aux champs de la boite de Petri */
void acboiti()
 {
   char respchamp, respboit, nomboit[4];

            printf("vous voulez sauver la boite de Petri (y/n)?  \n");
  for (;;) {
  scanf("%c", &respboit);
   if (respboit == 'y')
     {
       printf("\n entrer un identificateur de la boite, p.ex. YI:  ")
       scanf("%s", nomboit);
       printf("\n allez vous sauver le champ de celluels (y/n) ");
       for(;;)
        {
           scanf("%c", &respchamp);
           if (respchamp == 'n')
              break;
          else if (respchamp == 'y')
           savechampi(nomboit);
         }
       break;
      }
    }
 }

/*  La procedure de l'acquisition de l'image individuelle   */
acqind()
{

_clearscreen(_GCLEARSCREEN);
printf("**********************************************************
```

```
printf("* Bienvenue au programme d'acquisition d'images individuelles
printf("*************************************************************
acboiti();
printf("fin de l'acquisition individuelle");

}


  /***** compaucm.c *****/

 #include "prog.h"
 #include "pixlist.h"

extern int symsize;
/* extern char nomop[20];  */


/* #define MIN(a,b) (a < b) ? a : b
#define MAX(c,d) (c > d) ? c : d    */


int cellu, cellnoy, bruit;

void carre_image(x0, y0, dx, dy)
int x0, y0, dx, dy;

{
  int x, y;
  unsigned char gris0;

gris0 = 0;
for(y=y0; y < y0 + 10; y++)
   for (x = x0; x <= x0 + dx; x++)
putpixel(&gris0, x, y);
for (x = x0; x < x0 + 10; x++)
   for(y = y0 + 10; y <= y0 + dy; y++)
  putpixel(&gris0, x, y);
for(y = y0 + dy - 10; y < y0 + dy; y++)
   for (x = x0 + 10; x <= x0 + dx; x++)
        putpixel(&gris0, x, y);
for (x = x0 + dx - 10; x <= x0 + dx; x++)
    for(y = y0 + 10; y <= y0 + dy - 10; y++)
        putpixel(&gris0, x, y);
}


void fmaxmini(xm, ym, xa, ya)
int xm, ym, xa, ya;

{
   int minmax[4];
   FILE *fp2, *fopen();

  fp2 = fopen("maxmin.p", "ab");
  minmax[0] = xm;
  minmax[1] = ym;
  minmax[2] = xa;
  minmax[3] = ya;
  fwrite(minmax, sizeof(minmax), 1, fp2);
  fclose(fp2);

}

int danscontouri(xs, ys, xkc, ykc, k)
int *xs, *ys;
int k, xkc, ykc;

{
  unsigned char griss;
```

```
  switch( k )
  {
    case 0: {
              *xs = xkc + 1; *ys = ykc;
              getpixel(&griss, *xs, *ys);
            }
              break;

    case 1: {
              *xs = xkc + 1; *ys = ykc - 1;
              getpixel(&griss, *xs, *ys);
            }
              break;

    case 2: {
              *xs = xkc; *ys = ykc - 1;
              getpixel(&griss, *xs, *ys);
            }
              break;

    case 3: {
              *xs = xkc - 1; *ys = ykc - 1;
              getpixel(&griss, *xs, *ys);
            }
              break;

    case 4: {
              *xs = xkc - 1; *ys = ykc;
              getpixel(&griss, *xs, *ys);
            }
              break;
    case 5: {
              *xs = xkc - 1; *ys = ykc + 1;
              getpixel(&griss, *xs, *ys);
            }
              break;
    case 6: {
              *xs = xkc; *ys = ykc + 1;
              getpixel(&griss, *xs, *ys);
            }
              break;
    case 7: {
              *xs = xkc + 1; *ys = ykc + 1;
              getpixel(&griss, *xs, *ys);
            }
              break;
    }
    if (griss == 255) return(1);
    else return(0);

}

maxmini(xmi, ymi, xma, yma, xk, yk)
int  *xmi, *ymi, *xma, *yma;
int xk, yk;

{
  if (xk < *xmi) *xmi = xk;
  else
   if(xk > *xma) *xma = xk;

 if(yk < *ymi) *ymi = yk;
 else
   if(yk > *yma) *yma = yk;
}

deteccontouri(x0, y0)
int x0,y0;
```

```
{
  int xk, yk, xb, yb, xmin, xmax, ymin, ymax, k1, k2, first, found, c
 static int K;
 unsigned char grisk;

  xmin = x0; xmax = x0 + 1;
  ymin = y0; ymax = y0 + 1;
  cellnoy++;
  xk = x0; yk = y0;
  K = 6;
  first = 1;
  compp = 0;
  while (!((xk == x0) && (yk == y0)) || (first == 1))
{
  found = 0;
  while (found == 0)
 {
   if ((K - 1) < 0) k1 = 8 + (K - 1);
   else  k1 = K - 1;
   if ((K + 1) > 7) k2 = K + 1 - 8;
   else k2 = K + 1;
   if (danscontouri(&xb, &yb, xk, yk, k1))
   {
     xk = xb; yk = yb;
     K = K - 2;
     if (K < 0) K = K + 8;
     maxmini(&xmin, &ymin, &xmax, &ymax, xk, yk);
     found = 1;
     compp++;
   }
     else if (dans_contouri(&xb, &yb, xk, yk, K))
         {
           xk = xb; yk = yb;
           maxmini(&xmin, &ymin, &xmax, &ymax, xk, yk);
           found = 1;
           compp++;
         }
          else if (dans_contouri(&xb, &yb, xk, yk, k2))
                {
                 xk = xb; yk = yb;
                 maxmini(&xmin, &ymin, &xmax, &ymax, xk, yk);
                 found = 1;
                 compp++;
                }
                else {
                     K = K + 2;
                     if (K >= 8) K = K - 8;
                   }
    }
    first = 0;
  }
  fmaxmini(xmin, ymin, xmax, ymax);
  if (compp > bruit)
     cellu++;
 }

pointexpi(x, y)
int x, y;
{
FILE *fp2, *fopen();
int i, mima[4];

if ((fp2 = fopen("maxmin.p", "rb")) == NULL)
{
 printf("\n cannot open this file");
 exit(0);
}
 for(i = 1; i <= cellnoy; i++)
{
  if(!(fread(mima, sizeof(mima), 1, fp2)))
```

```
  if (feof(fp2))
    {
        fclose(fp2);
        exit(0);
     }
  else
     printf("file reading error\n");
if(((x >= mima[0]) && (x <= mima[2])) && ((y >= mima[1]) && (y <= mim
{
  fclose(fp2);
  return(1);
}
}
 fclose(fp2);
 return(0);
}

balayagei(x0, y0, dx, dy)
int x0, y0, dx, dy;

{
  int X, Y, i, j, firstcon;
  unsigned char gripix,bkgrd;

  Y=y0;
  bkgrd=0;
  firstcon = 0;
  modimagei(x0, y0, dx, dy);
  for(Y=y0; Y<y0+dy; Y++)
{
  X=x0;
  while (X < x0+dx)
{
  getpixel(&gripix, X, Y);
  if (bkgrd != gripix)
        if ( firstcon == 0 ) {
         deteccontouri(X, Y);
         firstcon++;
        }
        else
         if (!(point_expi(X, Y)))
         /*  printf("this point is not in the file, to ftrack\n"); */
            deteccontouri(X, Y);
    X++;
  }
 }
}

void choiseuili(x, y, dx, dy, seuilp)
int x, y, dx, dy, *seuilp;

{
  float FVAL[8];
  long HISTO[256], HIST;
  char SMIN,SMAX;
  int i;

  bimcopy(x, y, dx, dy, x + 512, y);
  disppos(40, 500);
  setmmlin(HISTO, 512, 0);
  flblkhst(x, y, dx, dy, HISTO, FVAL);
  disphfir(HISTO, -1);
  disppos(40, 500);
  bimcopy(x + 512, y, dx, dy, x, y);
  *seuilp = FVAL[3];
  printf("\nseuil de binarisation = %d", *seuilp);
}

 void traitimagei(x0, y0, dx, dy)
int  x0, y0, dx, dy;
```

```
{
  int seuil;
  char comment[60], cbim;

  choiseuili(x0, y0, dx, dy, &seuil);
  sobfir(x0, y0, dx, dy, x0, y0);
  seuil = seuil + 10;
  binfir(x0, y0, dx, dy, x0, y0, seuil);
  median(x0, y0, dx, dy, x0, y0);
  dilate2(x0, y0, dx, dy, x0, y0, 1);
  /* erode1(x0, y0, dx, dy, x0, y0, 1);*/
 }


  lectuimagei(x, y, dx, dy, fichnom)
  int *x, *y, *dx, *dy;
  char *fichnom;

{
  char comment[60], cbim;

  initfgb();
  presetfgb(0);
  DSKRDHDR(fichnom, comment, x, y, dx, dy, &cbim);
  diskread(fichnom, *x,  *y);
}


/* void size_fisym(fichsym, symnombre)
char *fichsym;
int *symnombre;
{
  struct symboly symbol;
  FILE *fp, *fopen();

  *symnombre = 0;
  if ((fp = fopen(fichsym, "rb")) == NULL)
{
  printf("le fichier %s n'existe pas !\n", fichsym);
  exit(0);
}
  while (fp != NULL)
{
  fread(&symbol, sizeof(struct symboly), 1, fp);
  *symnombre++;
}
  fclose(fp);
}  */


/****  Lectur des informations symboliques dans le fichier symb.p  **

void read_symb(sizesym, itemcour, fichsym, fichfim, typecell)
char *fichsym, *fichfim;
int sizesym, itemcour, *typecell;
{
  int i;
  struct dosdate_t date;
  struct symboly symbol;
  FILE *fp, *fopen();

  fp = fopen(fichsym, "rb");
  for (i=1; i <= sizesym; i++)
   {
    fread(&symbol, sizeof(struct symboly), 1, fp);
   if (i == itemcour)
  {
    strcpy(fichfim, symbol.nomfisym);
    *typecell = symbol.typecell;
```

```
    break;
   }
  }
   fclose(fp);
 }

/****   Trouver les informations concernant a l'image digitalisee ***

acqparami(nomfich)
char *nomfich;

{
  int  ok;
  char verf;

  ok = 1;
  verf = '0';
  while ( ok == 1 )
  {
    printf("\n  donnez le nom de fichier symbolique (4 car) " );
    scanf("%s", nomfich);
    printf("\n definir un critère pour éliminer les bruits ");
    scanf("%d", &bruit);
    printf("\n verifiez les entrées (n/y) ?");
    for(;;) {
    verf = getchar();
    if (verf == 'y')
        { ok = 1;
          break;
    }
    else if (verf == 'n')
        { ok = 0;
          break;
        }
    }
   }
 }

int veriffichi(fichnom)
char *fichnom;

{
  FILE *fp2, *fopen();

  if ((fp2 = fopen(fichnom, "rb")) == NULL)
  {
    fclose(fp2);
    printf("le fichier %s n'existe pas !\n", fichnom);
    exit(0);
  }
  else
  {
    fclose(fp2);
    return(1);
   }
  }


/************** imprimer les resultats sur l'ecran de video  ********

void  clear_monit(cellu, cellnoy)
int cellu, cellnoy;
{
  char rescom[30];

  effecran();
  strcpy(rescom, "Le resultat du comptage est :");
  curseur(20, 13);
  ecrire(rescom);
  curseur(20, 16);
```

```
  strcpy(rescom, "Il y a  ");
  ecrire(rescom);
  ecrirent(cellu);
  strcpy(rescom, "  cellules.");
  ecrire(rescom);

}

/*********************************************************
*    La procedure principale du comptage automatique      *
*********************************************************/

comtra_e()
{
  int x0, y0, dx, dy, i, typecell, symnombre;
  char nomfich[6], fichnom[9], fichsym[9];

 _clearscreen(_GCLEARSCREEN);
 printf("\n*********************************************************\n
 printf(" *     Bienvenue au programme de comptage automatique    *\n
 printf("*********************************************************\
 acqparami(nomfich);
 symnombre = symsize;
 strcpy(fichsym, nomfich);
 strcat(fichsym, ".p");
 printf("la taille du fichier %s est %d ", fichsym, symnombre);
 for (i=1; i <= symnombre; i++)
{
 read_symb(symnombre, i, fichsym, fichnom, &typecell);
 veriffichi(fichnom);
 cellu = 0;
 cellnoy = 0;
 printf("\n nom de fichier = %s",fichnom);
 lectuimagei(&x0, &y0, &dx, &dy, fichnom);
 traitimagei(x0, y0, dx, dy);
 balayagei(x0, y0, dx, dy);
 printf("\n\n*********************************************************
 printf("\n* Il y a %d cellules dans le fichier %s             * ",
 cellu, fichnom);
 printf("\n*********************************************************
 printf("\n %d points de bruits sont elimines\n", (cellnoy - cellu));
 clear_monit(cellu, cellnoy);
 system("DEL MAXMIN.P");
 }

}



/* *********************************************************

 *                Menu principal                           *
          Module d'interface interactive

 ********************************************************* */

#include "prog.h"

int symsize;
char nomop[20];

menu_princip()
{
  short choix;
  choix=0;
  while (choix != 5)
      {
      system("cls");
              printf("\n     MENU PRINCIPAL DU COMPTAGE DES CELLULES
      printf(" ----------------------------------------------\n\n
```

```
      printf("   ACQUISITION DE L'IMAGE INDIVIDUELLE          (1)   \n\
      printf("   ACQUISITION INTERACTIVE D'IMAGEs             (2)   \n\
      printf("   COMPTAGE DES CELLULES D'UNE IMAGE            (3)   \n\
      printf("   COMPTAGE AUTOMATIQUE DE CELLULES D'IMAGE     (4)   \n\
      printf("   quitter                                      (5)   \n\
      printf("choix ?  ");
      scanf("%d",&choix);
      switch(choix)
            {
              case 1: acqind();   break;
              case 2: acboit_e(); break;
              case 3: compind();  break;
              case 4: comtra_e(); break;
              case 5: break;
            }
      }
}

/* ****************************************************************

 *                       le programme principal                    *

 ****************************************************************** *

main()
{
  _clearscreen(_GCLEARSCREEN);
  symsize = 0;
  printf("\nVoulez-vous donner votre nom S.V.P  ");
  for (;;)
     if ((gets(nomop)) != NULL)
       {
       menu_princip();
       break;
       }
   /* system("DEL symb.p); */
   printf("fin de la program\n\n");
}
```